

University of Alabama in Huntsville

LOUIS

Honors Capstone Projects and Theses

Honors College

4-27-2023

Using Big Data Analytics for Sentiment Analysis on Team Communication and Situational Awareness

Kyle Robert Fletcher

Follow this and additional works at: <https://louis.uah.edu/honors-capstones>

Recommended Citation

Fletcher, Kyle Robert, "Using Big Data Analytics for Sentiment Analysis on Team Communication and Situational Awareness" (2023). *Honors Capstone Projects and Theses*. 797.
<https://louis.uah.edu/honors-capstones/797>

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.

Using Big Data Analytics for Sentiment Analysis on Team Communication and Situational Awareness

by

Kyle Robert Fletcher

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 24th, 2023

Honors Capstone Director: Dr. Vineetha Menon

Associate Professor Computer Science



04/27/2023

Student

Date



04/28/2023

Director

Date

Department Chair

Date

Honors College Dean

Date



Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu


Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Kyle Fletcher

Student Name (printed)



Student Signature

04/24/2023

Date

Table of Contents

Copyright Permission	pg. 1
Table of Contents	pg. 2
Dedication	pg. 3
Abstract	pg. 4
Introduction	pg. 5
Methodologies	pg. 6
Results and Analysis	pg. 12
Conclusion	pg. 22
References	pg. 23
Appendix	pg. 24

Dedication

This Honors Capstone is dedicated to Xeen, the love of my life and the person that kept me motivated to complete this work.

Abstract

Team communication is incredibly important to team performance, and situational awareness is a large part of successful team communication. This study utilizes team communication data from multiple teams in a firefighting simulation and performs data analysis on it. The data from the simulation contains situational awareness ratings, which are used to split up the data into different categories of performance in the aspect of situational awareness. Then multiple types of feature extraction are done on the data, most of which involve some form of natural language processing. The features extracted also go through a process of dimensionality reduction before they are combined and used to train AI/Machine Learning classification algorithms. These algorithms attempt to place teams and individuals from the simulation into the correct category corresponding to their performance in the aspect of situational awareness ratings. This study dives into what the different features extracted from the communication data look like and how they contribute to our understanding of the data and the machine's understanding of the data. The results of running the AI/Machine Learning algorithms are also analyzed to quantify how well the extracted features allow a machine to predict levels of situational awareness.

Introduction

In all situations, and especially high stress ones that involve teams, the communication abilities between team members are especially important. Generally, the quality of communication within a team has real effects on a team's performance.[1] Now more than ever, there are more and more cases where machines are present within team situations. Therefore, there are many opportunities to make use of machines to improve the communication that goes on within teams. In order to do this, machines must understand the aspects of team communication.

One very important aspect of communication is situational awareness. Situational awareness is especially important in higher stress scenarios such as combat or emergency situations where all team members need to properly communicate what is going on. Situational awareness refers to how much members of a team understand the environment they're in and the changing state of the environment over time. The level of situational awareness conveyed through team communication does impact the success of a team in many situations.[2] It stands to reason that if a machine that can understand the concept of situational awareness, teams that involve human machine interaction can benefit from the machine's understanding and improved communication.

The first step of developing a machine's understanding of situational awareness is seeing if it can identify it. That brings us to the scope of this study. If we gather communication data from teams in a simulated high stress environment, can a machine be trained on that data to identify how situationally aware the teams in the environment are? For the scope of this study the levels of situational awareness among different teams of people will have been already generated, and we will test how well a machine can process the communication data along with the already known situational awareness levels and determine in future communications how situationally aware a team is.

This study specifically is based on a previous study "Using Big Data Analytics for Sentiment Analysis to Explore Team Communication Dynamics in Human Machine Interactions for Team Situational Awareness".[2] The goal of this study is to improve upon that study by investigating more avenues of data analysis on team communication data, specifically ones related to natural language processing. This study evaluates how the newly investigated features of communication data can be utilized to improve the classification of situational awareness by a machine. This study also aims to go into as much detail as possible about the various ways it analyzes team communication data.

Methodology

Simulation Details

The dataset used is the same as in “Using Big Data Analytics for Sentiment Analysis to Explore Team Communication Dynamics in Human Machine Interactions for Team Situational Awareness”.[2] It comes from the same simulation where teams of four members carried out simulated fire-rescue operations. The goal of the teams was to put out forest and building fires. These fire-rescue operations were designed to test the team’s collaboration skills. The simulation is described sufficiently in the paper referenced.

All team members were assigned independent workstations with a single computer. Each team members was given role and task-specific information prior to the start of the fire-rescue exercise. The teams interacted with one training simulation (simulation 0.0). This was followed by two independent fire rescue operations: simulation 1.0 and simulation 3.0, respectively, that lasted 15 minutes each. The primary task in each of the simulations was to protect the forest and housing artifacts in the environment from fire eruptions and control the fire eruption situations in the environment. All team members had full visibility of the simulation environment assets such as the terrain, housing, forest and water towers to refill the water tank systems. The simulation 2.0 was a team discussion session that was held in between the fire rescue simulations 1.0 and 3.0, respectively...Each team member was assigned their own fire response system, either a fire-engine or helicopter. Each fire-response system had a different capability such as speed or limited water holding capacity...Throughout the simulation all teams were able to send and receive information in text-based format using the chat window option [2]

The data produced from this simulation that we will use is based upon what was recorded from the chat window during these simulations. This paper does not investigate any other data produced by the simulation.

Data Details

The data consists of messages from each team’s chat during the simulations they participated in. The messages from each member were recorded along with some other identifying information. Since there were 4 members in each team, each message is paired with a number 1 through 4 identifying the individual who posted it. The message is also paired with the simulation number 0 through 3 and the time within that simulation that the message was sent. Finally, some messages have a situational awareness score or two paired with them which will be described shortly. Each of the 41 teams had all their messages along with this information recorded. Figure 1 shows how the data looks in excel format, which is the baseline from which this project is built. The figure shows columns “Turn Taking”, which designates the team member which sent the message, “Time”, which is the time that the message was sent, “Simulation”, which is the number simulation that the team was in when the message was sent, and “Team SA” which is the Situational Awareness Score/Scores for the message.

1	Turn Taking	Time	Simulation	Team SA
18	4	0:07:20	that is all i see	0
19	3	0:07:32	what third should i get	0
20	4	0:08:13	i say we just go free for all	6
21	3	0:08:22	sweet	0
22	1	0:08:22	agree	0
23	3	0:09:12	we are not to good	0
24	1	0:09:38	so there are only 3 of us?	0
25	2	0:10:00	there's four but my computer isn't wanting tocooperate so i just get to watch for fires	0 6,3
26	1	0:10:00	that was so fun	0
27	3	0:00:00	hey	1
28	2	0:00:00	well hi there!	1
29	3	0:00:00	where is your location	1
30	4	0:00:20	lets divide this. how many of us are there?	1
31	3	0:00:23	ill get lower portion	1 6,4
32	1	0:00:24	ill go bottom right	1 6,4
33	3	0:00:30	lower left	1

Figure 1: Spreadsheet View of the Simulation Data

The numbers in the right column are the Situational Awareness Scores, or SA scores for short. Two domain experts are responsible for generating these scores per message. The two experts came to a complete agreement on each score. The scores range from 3 to 6 and each represents a different level of situational awareness. A score of 3 corresponds with perception. A score of 4 corresponds with comprehension. A score of 5 corresponds with projection. Finally, a score of 6 corresponds with Action. Table 1, taken from the paper mentioned before that described these simulations, displays these situational awareness scores with their actual definition and an example.[2]

Code #	Team Situation Awareness	Definition	Examples
3	Perception	Information about team factors and their current state such as condition, modes, action.	"My water tank is empty"
4	Comprehension	Information about task related occurrences within the team that help other team members understand team relations, team events and places.	"Our fire engines seem to be slower than our helicopters"
5	Projection	Information about possible future actions in regards to own team.	"We will have to split up areas"
6	Action	Action of team members in regards to own team.	"Fire engines fill up your water tank"

Table 1: Situational Awareness Score Categories along with their Definition and Examples (Adapted from [2])

Data Cleaning

With this data the first necessary step to take was to clean it. The cleaning process involved a few simple steps. First the data was converted from its excel format to python Pandas

data frames. These data frames were used to contain only the columns that were meant to be there and with no empty rows. Once this was done, each team's and each team member's total number of messages were obtained and added onto the data frames. This was done so that we could have an accurate count of these values before the next step in the cleaning process. The next step involved finding every row that was given 2 SA scores and separating the two scores. From here the second score was added with its message and other data as a new row to its team's data. Since the count of messages wouldn't be the real count after this step those counts had to be recorded before it even though that isn't normally part of the cleaning process. This completed the data cleaning process and then it was time to categorize the data.

Categorization of Data

The next step to take with this data was to categorize it into different performance classes. Performance in this case means how situationally aware each team and each team member were. For the purposes of this project both the situational awareness of the team overall, and the situational awareness of each individual were considered. For categories, both the teams, and total individuals in the data set were split into 3 groups of equal size for low, mid, and high performance. In order to split up the teams and individuals a metric for the performance was needed. For this the mode situational awareness score and two averages were summed up. The first average was the average situational awareness among all the messages that were actually scored, and the second was the average situational awareness among all the messages, including the ones that were not given a score. With the sum of these values as a performance metric, the teams and individuals were split into their categories accordingly. There were 14 low performance teams, 14 mid performance teams, and 13 high performance teams, similarly, there were 56 low performance individuals, 56 mid performance individuals, and 52 high performance individuals.

Data Preprocessing

In order for a number of the data features that were used in the data analysis for this project to be properly extracted, the raw text data from each message had to go through some preprocessing. Firstly, all the text in each message was made lowercase. This is to avoid upper and lowercase of the same word registering as 2 different words. Next the text data was tokenized using NLTK's word tokenizer. This split the raw text into arrays of the individual words. After that the words were all lemmatized using NLTK's wordnet lemmatizer. This step reduces most words to their base form. For example the words "liking" and "liked" would be turned into "like". After this the set of English stop words from NLTK along with common punctuation was removed from the arrays of text. Stop words are super common words like "the" and "a" that don't offer much meaning to a set of text. Because they tend not to offer much meaning they were removed. Finally, every individual word was only kept if it was found to be a real word in the NLTK words corpus. This was done to remove misspellings and words that aren't real words. [3] All of these steps make of the data preprocessing for the text. This preprocessed text data is later used for both the TF_IDF features and the part of speech features, but it not used for the sentiment intensity features.

Feature Extraction and Natural Language Processing

The next thing to do was extract features from the data for the data analysis. Most of the features that were extracted from the data are related to natural language processing, though not all are.

First off are the features that aren't based on the text of the messages. These features are all related to the number and frequency of messages sent. Specifically, for each team and for each individual, the number of total messages sent by that team/individual was a feature. The frequency of messages sent by a team/individual in a single simulation also made up 4 more features since there are 4 simulations for messages to take place in and have a frequency for. This made 5 total features based on the number and frequency of messages sent.

The next set of features comes from part of speech tagging. Part of speech tagging involves classifying each word in a body of text as being a particular part of speech. For this task NLTK's part of speech tagger was used to determine the part of speech for each word.[3] The tags generated by this were used to create a matrix of each part of speech with the average amount of times that part of speech showed up in a message for all the messages from a team, and a separate matrix with the same properties but with all the messages from an individual. This produced roughly 27 features each being the various parts of speech found in the body of text.

The next text-based features come from TF_IDF being performed on all the text from each team and each individual. TF_IDF is similar to bag of words in that it generates a sparse matrix with values corresponding to whether each word out of the entire set of words in the data exists in a particular team/individual's document, where document refers to the set of words generated by that person or team. However, where bag of words only counts the occurrences of the word in the document, TF_IDF does a bit more complex of a calculation. It takes into account how much the word appears in the document multiplies it by how little the word appears in other documents. Therefore TF_IDF scores are maximized when a word both occurs in a document and doesn't occur in other documents. The actual formula for this score calculation is as follows.[4]

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

$$IDF = \log \left(\frac{\text{number of the documents in the corpus}}{\text{number of the documents in the corpus containing the term}} \right)$$

$$TFIDF = TF * IDF$$

TF_IDF was chosen since it should help further separate the different documents and make classifying them easier than it would be with only bag of words. Scikit-Learn's TF_IDF vectorizer was used in this case to generate the sparse feature matrix, which in total contains a feature for each unique word in the complete set of all text from the simulation, which made up around 1700 features.[5]

The final set of features that were used in the data analysis were related to sentiment analysis of the text. Sentiment Analysis refers to determining how positive, negative, or neutral a body of text is. In order to accomplish this task, the valence aware dictionary for sentiment reasoning (VADER) model in NLTK was used. This model has a sentiment intensity analyzer which, given a body of text, can generate a positive sentiment rating, neutral sentiment rating, negative sentiment rating, and compound sentiment rating based on the previous 3 ratings.[6] In

this case the sentiment intensity analyzer was passed each message from a team or individual to generate these values for. Then the values for each of these ratings were averaged across all of the messages from a particular team or individual in order to generate the sentiment features for them. With this method, there were 4 features generated for each team and individual, being the average of each sentiment rating.

Dimensionality Reduction

With the feature extraction being done we end up with 5 message count/frequency based features, 27 part of speech based features, ~1700 TF_IDF based features, and 4 sentiment analysis based features. In order to best use these features to train classifier models, some form of dimensionality reduction is necessary. For this project Linear Discriminant Analysis, or LDA was used. LDA can be used to reduce a set of features down to any amount of dimensions from 1 to the total number of classes – 1 dimensions. This is because LDA is a supervised learning method and takes into account the actual classes of the data. Since there are 3 performance classes in our data the choice for number of dimensions to reduce to was either 1 or 2. For each set of features described previously, LDA was performed on it to reduce it down to 2 dimensions. What LDA is meant to do is, within the dimensions the data is reduced to, minimize the separation between data points of the same class, and simultaneously maximize the separation between data points of different classes. Using this method, ideally, the data becomes more easily separable for the classifiers to more accurately classify it. For this project scikit-learn's Linear Discriminant Analysis module is used to perform LDA on each set of features, bringing the total features down to 8, 2 features for each of the 4 sets of features.[5] With this step the architecture of this projects data analysis becomes clear. The natural language processing-based AI/ML architecture is displayed below in figure 2.

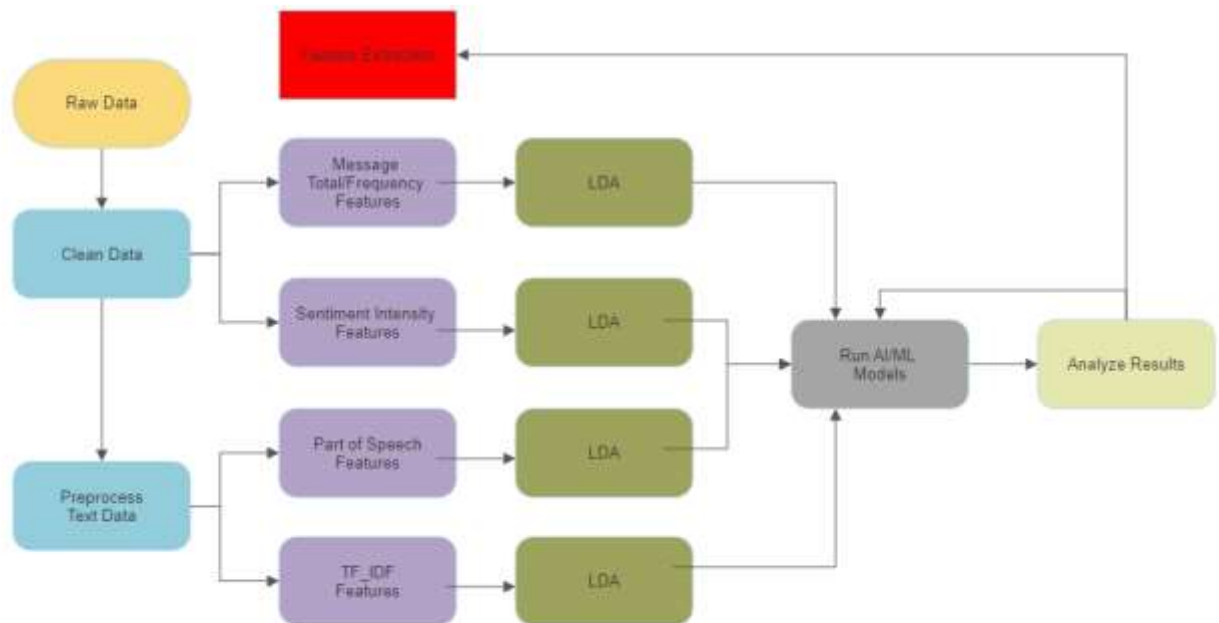


Figure 2: Architecture for Predicting Team Performance from Derived Features of the Data.

AI/ML Models

Now that we have 4 sets of 2 features dimensionally reduced, we can test if they can successfully train various classifiers. For this project 7 different classifiers were used. These classifiers were trained on both the data categorized by each team and the data categorized by each individual member.

3 of the 7 classifiers were various types of Support Vector Machines, or SVMs. Support Vector Machines are a classifier that attempts to find the decision boundary between classes of data that has the widest margin between the multiple classes. One type of SVM is linear SVM and involves the decision boundary being a straight line. However, the kernel in SVM can be changed so the decision boundary doesn't have to be linear which can better classify certain data. Both the polynomial (Poly) kernel and the RBF kernel can be used with SVM to generate non-linear decision boundaries.[7] For this project, all 3 of Linear SVM, Poly kernel SVM, and RBF kernel SVM were used, each with a C constant value of 3. Specifically, Scikit-Learn's implementation of SVM in these 3 ways was used to implement it.[5]

Another classifier used is the Naïve Bayes classifier. Naïve Bayes uses Bayes Theory and attempts to choose the class that maximizes the posterior probability of the class being correct given the document. It uses the likelihood, the probability of a document given the class, along with the probability of the class to calculate the posterior probability. [2], [8] Since we did LDA on all of our sets of features, we have all continuous type data and therefore simple Gaussian Naïve Bayes is the type we chose use for our classifier. For the implementation Scikit-Learn was once again used to perform the Naïve Bayes classification.[5]

The next classifier used is the Multilayer Perceptron, or MLP. MLP is an AI neural network algorithm that takes the features for classification at its input layer and generates the classification at its output layer by applying weights learned through training. It's called multilayer because there can be any number of hidden layers in between the input and the output layer.[2] One benefit is adding hidden layers allows the model to adapt to non-linear behavior. [9]. For this project again, Scikit-Learn's MLP Classifier was used with its default settings, which involve 100 hidden layers and using the adam solver.[5]

Another powerful classifier that was used are Decision Trees. Decision trees create a tree structure of feature-based decisions that all go into determining the class of a document which is done on the leaf nodes of the tree. Once a tree is trained the input features of a document will go along a path of the tree where each branch considers the value of a feature until it reaches a leaf node which will determine what class the document is classified as.[2], [10] For this project, Scikit-Learn's Decision Tree Classifier was used to run this model and was done with the default settings.[5]

The final classifier used is one that is based on Decision Trees and is known as Random Forest. Random forest is essentially just an ensemble of Decision Trees performing together to get a more accurate classification. An ensemble is when a number of estimators are used together to provide a better result than the estimators by themselves could. In the case of Random Forest, an ensemble of Decision Trees is made using bagging, which means the Random Forest classifier chooses whatever class the most Decision Trees in the ensemble classify the object as.[2], [10] For this project, once again, Scikit-Learn's Random Forest Classifier was used to run this model

and it was set to the default values which generate 100 different Decision Trees as estimators to use in the classification.[5]

Training Models and Validation

We have the classification models and the features to input into them but its important to discuss how the models are trained and validated. First of all, the random seed that's put into splitting a dataset into a training and testing set can affect what classification accuracy ultimately ends up getting outputted since a change in the makeup of training and testing sets will inevitably affect how the classifiers learn and classify the data. In order to counteract this, this project used a method of validation known as a Stratified Shuffle Split. The Shuffle Split part of that just means that the data is shuffled and split into training and testing data a specified number of times to counteract the variance of outcomes caused by a single random seed. Then the Stratified part of that means that the train and test sets have an equal proportion of all the classes split up among them. For this project Scikit-Learn's Stratified Shuffle Split implementation was used with $n = 5$ meaning 5 shuffled test sets and train sets were created for application to the classification models.[5] The models were tested even further since different training set sizes were passed to the Stratified Shuffle Split. 30%, 40%, 50%, 60%, and 70% training data percentages were used with the Stratified Shuffle Split in order to truly validate how well each of the models were able to perform based on a range of training set sizes.

Results and Analysis

Data Categorization

After the data had been cleaned, it could be categorized into its performance classes. First for the categorization of teams, 14 teams were categorized as low performance, 14 were categorized as mid performance, and 13 teams were categorized as high performance. Figure 3 shows the total SA score makeup of the low performance class and the high performance class of the teams.

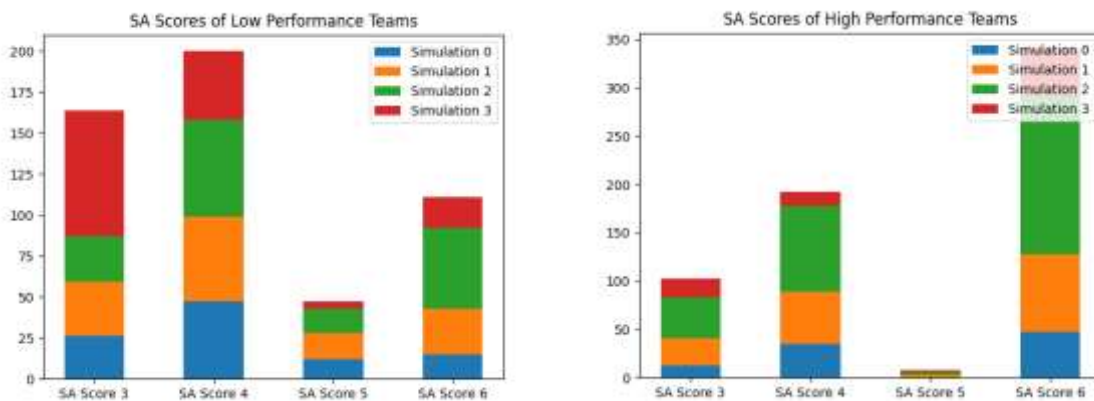


Figure 3: SA Score Makeup of Low and High Performance Teams

The biggest noticeable difference between the low performance teams and the high performance teams is the amount of SA scores of 6. The high performance teams actually had less scores of 3 and 5 but there were over 3 times as many scores of 6. This does make sense given that the mode is a factor in the performance calculation. Another observation is that in the high performance graph, most of the SA scores come from simulation 2, the team discussion portion. So even though the team members weren't fighting fires in that moment, the utilization of the team discussion is what made a large difference in their scores.

The individual team members were also categorized into low, mid, and high performance classes, with there being 56 team members in the low performance class, 56 team members in the mid performance class, and 52 team members in the high performance class. Similar to Figure 3, Figure 4 shows the total SA scores for the low performance and high performance classes of individuals.

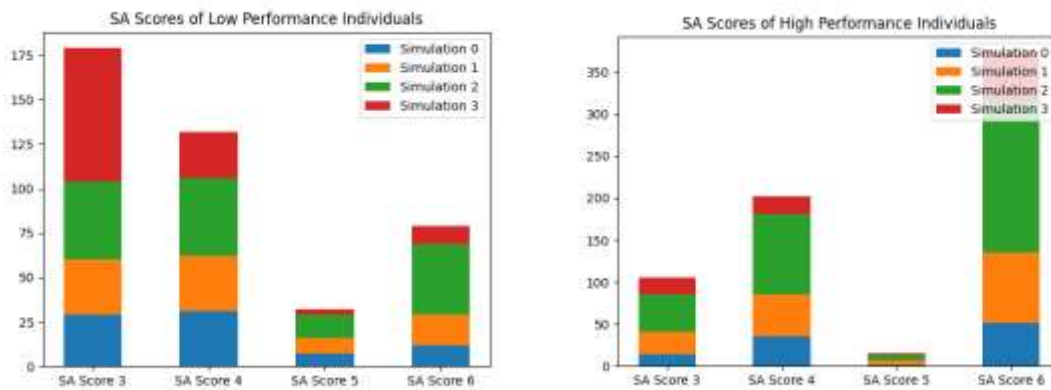


Figure 4: SA Score Makeup of Low and High Performance Individuals

The makeup of these classes of individuals closely mirrors the makeup of the classes of teams, with the largest difference between low and high performance being the amount of SA scores of 6, and simulation 2 almost always making up the majority of the SA scores.

Turn Taking Based Features

As stated in our methodology, 5 features were extracted from the data that were based on message total and frequency. These features were the total messages sent, the frequency of messages sent in simulation 0, the frequency of messages sent in simulation 1, the frequency of messages sent in simulation 2, and the frequency of messages sent in simulation 3. These features were extracted from both individuals and teams. Figure 5 shows the average amount of messages sent between the 3 classes of teams and 3 classes of individuals.

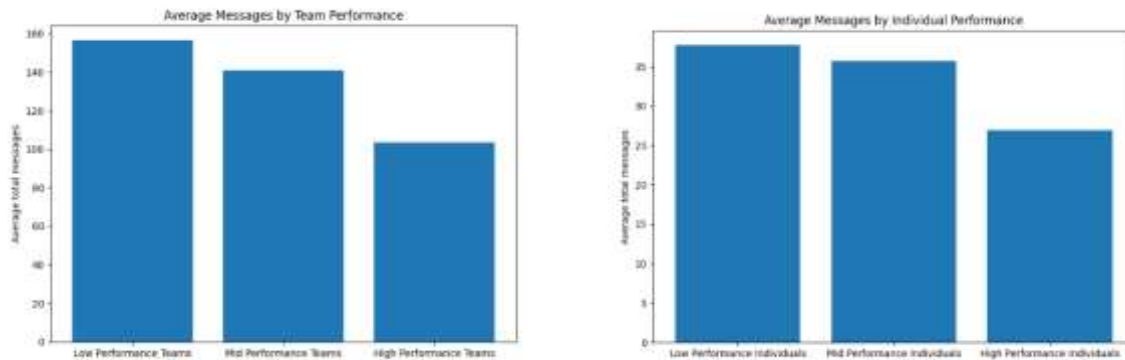


Figure 5: Average Total Messages for Different Performance Classes of Teams and Individuals

The clear trend here is that the teams with higher scores for SA generally sent less messages. The same trend is exhibited in the individual performance classes. This is partly due to how the performance was scored using averages but also could be the result of team members taking more time to write out situationally aware messages and not just sending lots of messages that have no importance.

Once we had this message total feature along with the 4 other message frequency features, all were normalized with Scikit-Learn's Standard Scaler and then LDA was applied to them to reduce them down to 2 components.[5] Figure 6 shows scatter plots with the LDA components as axis and the color as the actual class of the data for both the team and individual classes respectively (team on the left, individual on the right).

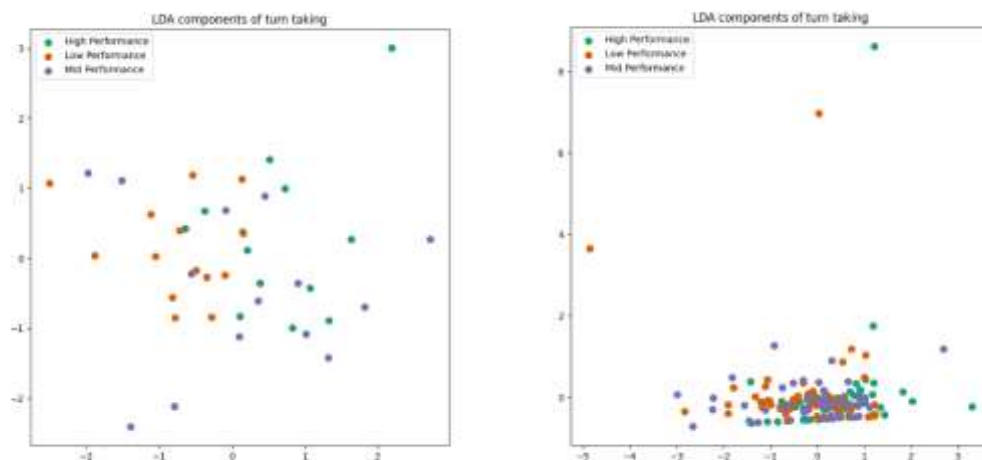


Figure 6: LDA Components of Turn Taking Features for Teams (left) and Individuals (right)

The LDA components of neither the team nor the individual classes are particularly separable. The team LDA components do show some slight differences in the outliers between the different classes. The individual LDA components, however, do not show much of a pattern at all, besides high performance teams tending to be higher on the x axis. These LDA components are 2 of the features that go into the classifiers later.

Part of Speech Features

The next set of features to be extracted were the ones from part of speech tagging. In this case, NLTK's part of speech tagger tagged each word of the preprocessed text data as 1 of roughly 27 parts of speech.[3] This produced 27 features for both the team performance and individual performance classes, with each feature being the average amount of times that part of speech appeared in a message sent by the team or individual. No visualization was done on this feature data until after LDA was applied. The features were again normalized with Scikit-Learn's Standard Scaler.[5] Then, LDA was applied to the part of speech features for both the team and individual data. Figure 7 shows the scatter plots of the LDA components for team and individual classes respectively (team on left, individual on right)

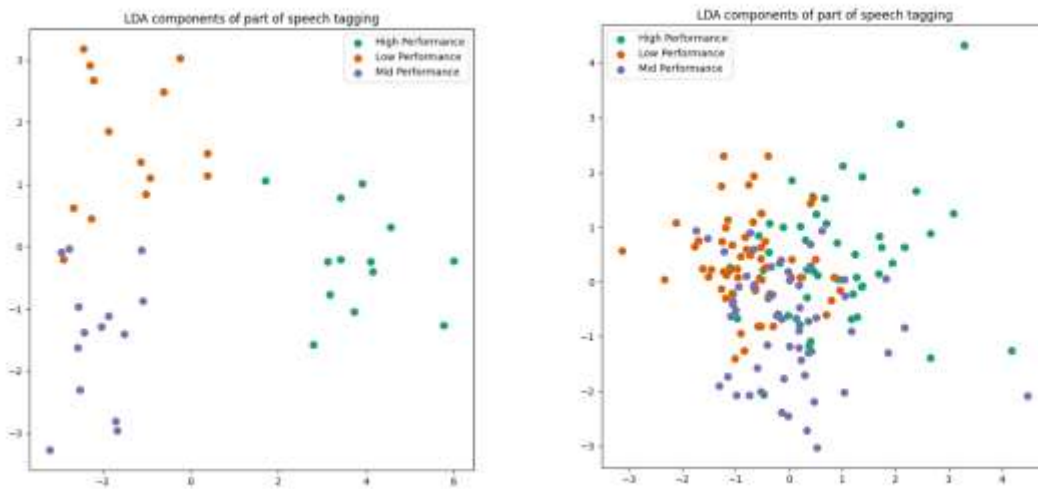


Figure 7: LDA Components of Part of Speech Features for Teams (left) and Individuals (right)

From the team plot of the LDA components, the classes are very clearly separable. It's pretty close to perfectly separating them, making the part of speech features very important for the team performance classification. On the other hand, the individual plot of LDA components does not separate the classes nearly as well, although it's still clear that the 3 classes have different areas in the plot. This means while the part of speech features are still important for the individual performance classification, they don't hold as much importance as for the team performance classification.

TF_IDF Features

As described in the methodology, TF_IDF was used on both the total text from each team and the total text from each individual to generate large feature matrices where the features were the TF_IDF scores for each word out of the whole set of text. This produced roughly 1700 features for both the team and individual classes of data. The first thing we did with this was find the 20 feature words with the highest TF_IDF score across the low and high performance classes for both teams and individuals. Figures 8 and 9 show the total of the 20 highest TF_IDF scores among low performance and high performance teams respectively.

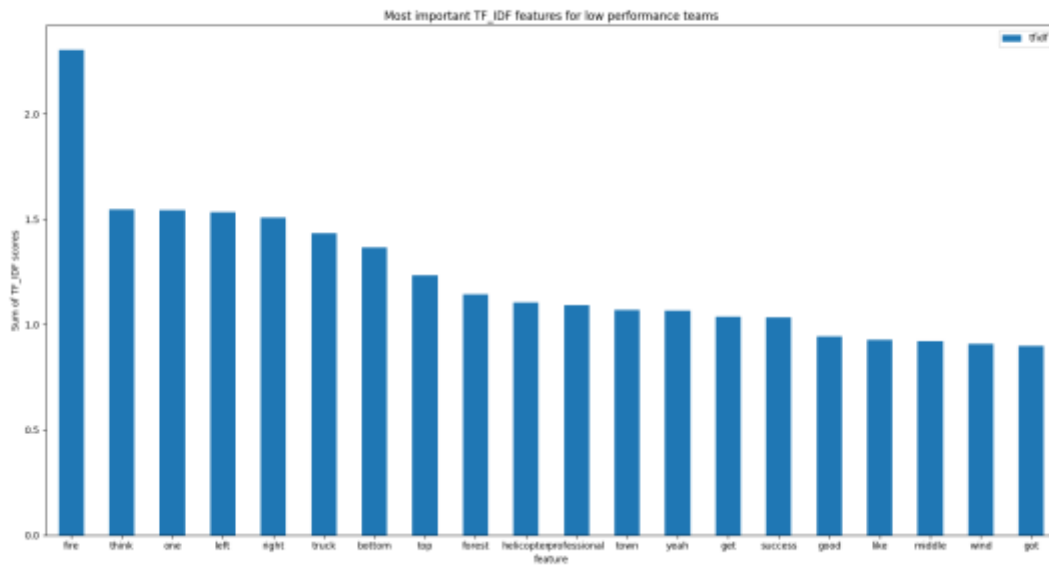


Figure 8: 20 Highest totals of TF-IDF Scores for Low Performance Teams

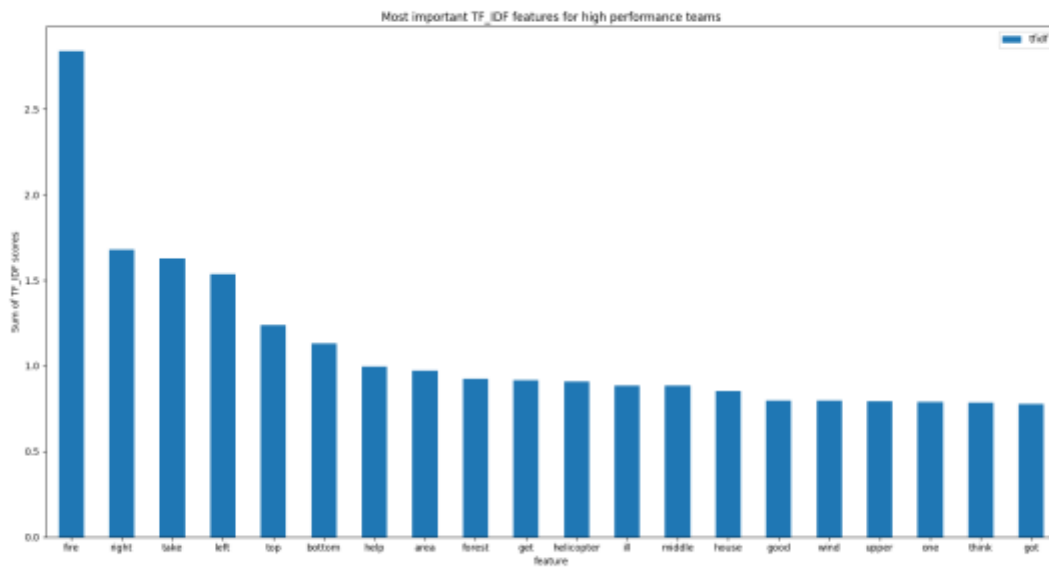


Figure 9: 20 Highest totals of TF-IDF Scores for High Performance Teams

For both of these we see a similar distribution of words although there are some key differences. First of all, “Fire” is just a super common word in this simulation regardless of the situational awareness of teams. There are a few words high up on the feature list for high performance that aren’t in the top 20 features for low performance teams. “Take”, and “Help” are the two main

examples of this, and it makes sense why as these are words that really show action, which has an SA score of 6.

We also generated graphs of the 20 most important TF_IDF features for the low performance and high performance individuals. These are shown in Figures 10 and 11 respectively.

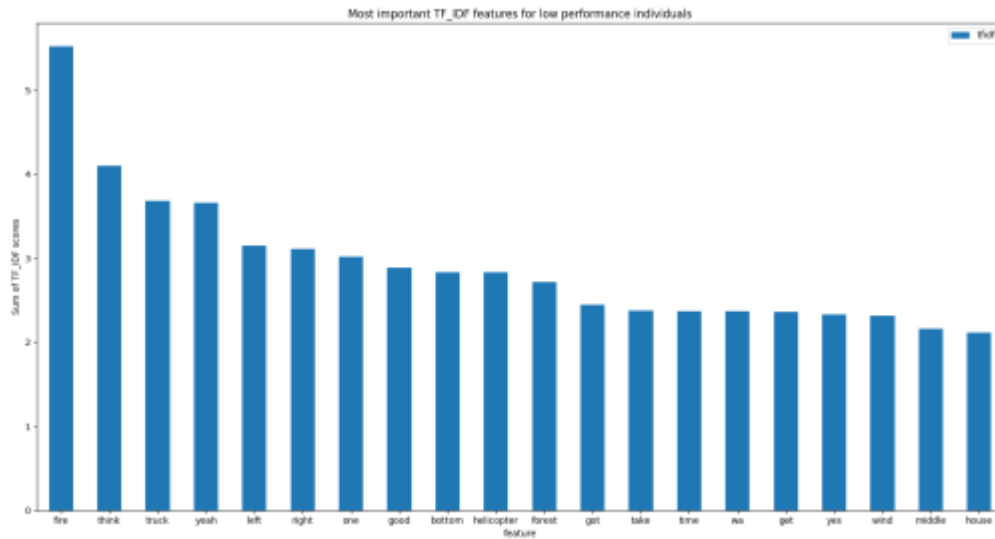


Figure 10: 20 Highest totals of TF_IDF Scores for Low Performance Individuals

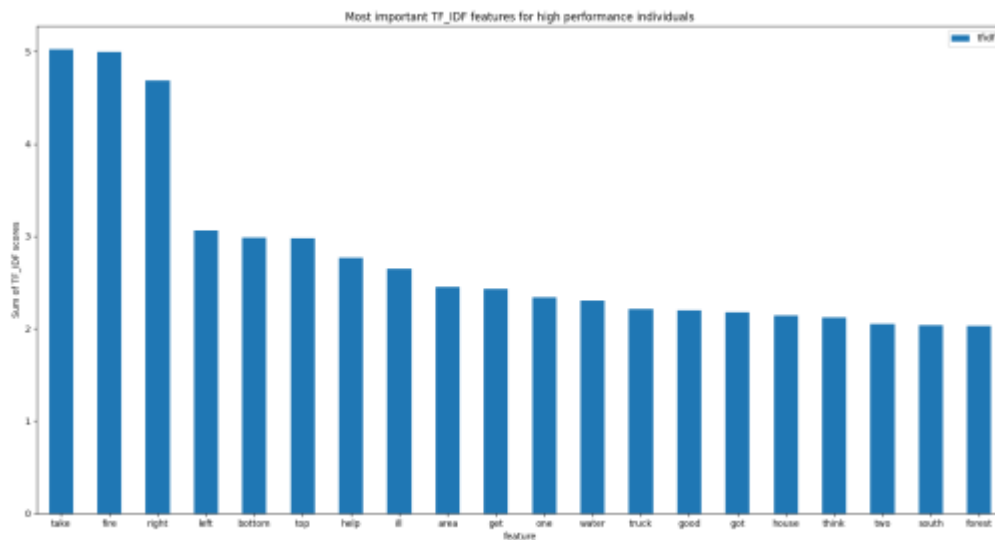


Figure 11: 20 Highest totals of TF_IDF Scores for High Performance Individuals

These graphs are not exactly the same as the ones for the team performance, however they show very similar trends and the word makeup between the two are similar. One thing to note is for the high performance individuals “Take” was so important of a word that it overtook “Fire”.

Once this data was visualized, it was time to perform LDA on the roughly 1700 features generated by TF_IDF to bring the total features down to 2. This again was done on both the team data and individual data. Figure 12 shows the 2 LDA components and their classes in scatter plots for both the Team and Individual performance classes respectively (team on the left, individual on the right)

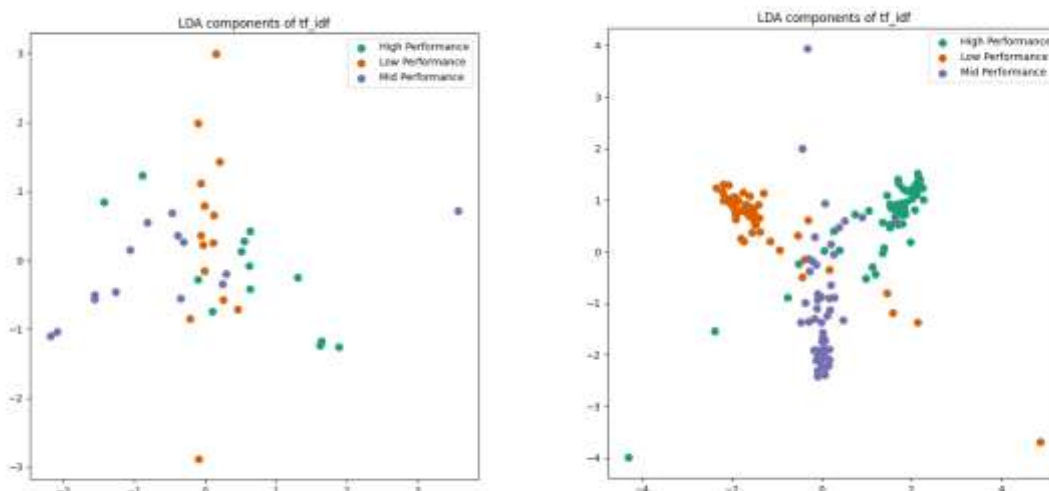


Figure 12: LDA Components of TF_IDF Features for Teams (left) and Individuals (right)

The team LDA component plot shows that the 2 components do not do a good job of separating the classes. However, the different classes do have slightly different shapes. On the other hand, the LDA component plot for the individual classes shows that the LDA components for those classes successfully separate most of the data points within each performance class. It's far from perfect but it looks like something a classifier could learn. What this means is that the components of TF_IDF for the individual performance classes are very important to the overall performance classifier while the TF_IDF components for the team performance classes are not very important to that overall performance classifier.

Sentiment Features

The final set of features extracted came from sentiment intensity scores generated by the Valence Aware Dictionary for Sentiment Reasoning (VADER) model in NLTK. There were 4 features generated. These were the averages of positive, negative, neutral, and compound sentiment polarity scores generated by VADER. These were averages since each individual message was passed in without preprocessing to get a score for all four categories. We didn't do any data visualization of these average sentiment scores until after LDA was performed. LDA was used to bring the 4 features down to 2 and attempt to make the classes more separable through the 2 LDA components. Figure 13 shows scatter plots of the 2 LDA components and their corresponding performance classes for both the team data and individual data respectively (team on left, individual on right)

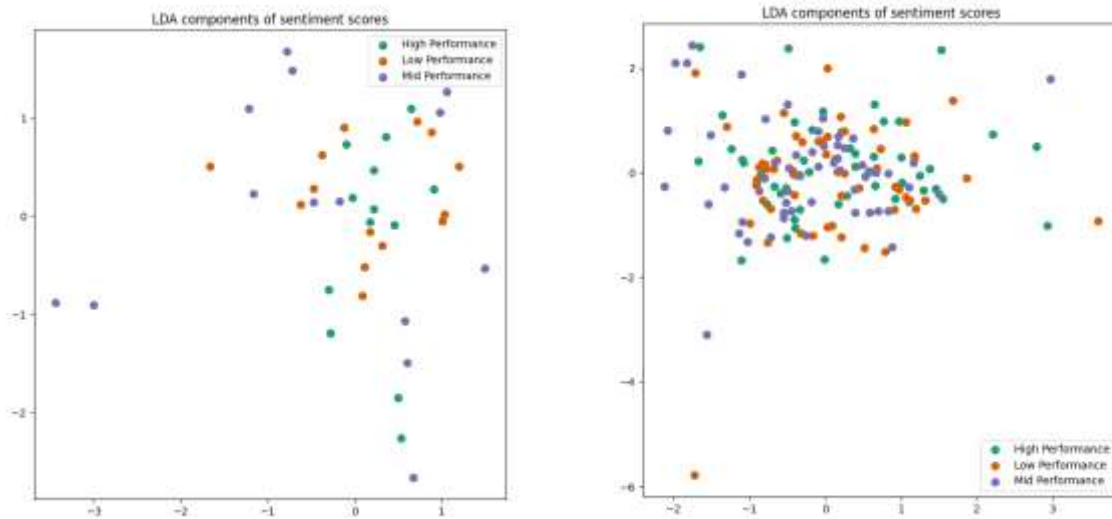


Figure 13: LDA Components of Sentiment Based Features for Teams (left) and Individuals (right)

These LDA scatter plots show that for both the Individual Performance classes and Team performance classes sentiment intensity paired with LDA does the worst out of all the features at separating classes. For the team performance classes the only clue given as to which class is which is the component variance of the class, since low performance takes up much less area on the plot. As for the individual performance classes there really aren't any clear observations that can be made about the differences between the classes. The result of this is the sentiment intensity features in our model will likely not be very helpful to the overall classifier for both team performance and individual performance.

Running the Models

Every step up until this point has been setting up for this stage of the analysis. Here we ran each AI/ML model described before on our set of features produced through LDA of different feature types, and observed how well we could classify both team and individual performance of data. A Stratified Shuffle Split was used to generate 5 random training and testing sets and each of these was ran through all 7 models to get classification accuracy scores. This was repeated with a train size of 30%, 40%, 50%, 60%, and 70%. Figure 14 shows a graph and table of the classification accuracy for each model for each training size for team performance.

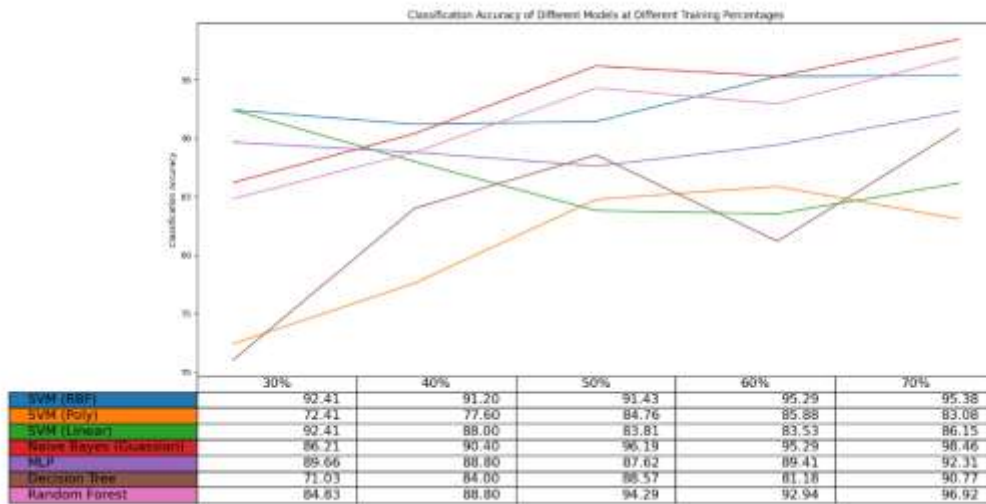


Figure 14: Team Performance Classification Accuracies by AI/ML Model and Training Size

It turns out the team classifiers performed quite well here, with accuracies ranging from 71% accuracy at the lowest to an incredible 98% accuracy at the highest. It's also a good sign that for every classifier besides linear support vector machines, the accuracy trended upward with an increased training size, meaning the more data it had to train from, the better it fit the data. Overall, it seems Naïve Bayes performed the best, being the model that gave us the highest classification accuracy we saw.

After running the models on the features from the team data, it was time to run them on the individual data and see how accurately individual performance could be classified. The same method used on the team performance was used for individual performance. Figure 15 shows a graph and table of the classification accuracy for each model for each training size for individual performance.

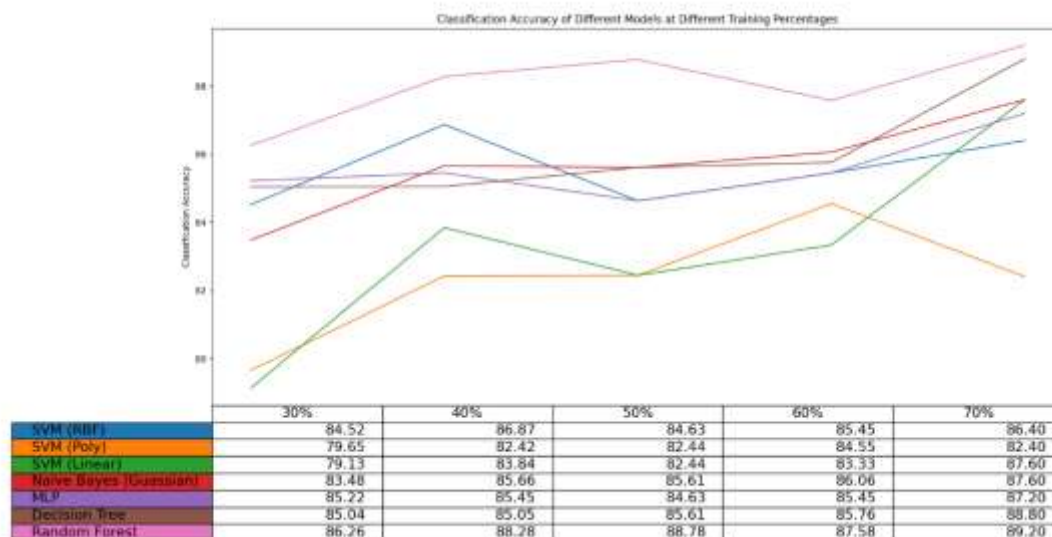


Figure 15: Individual Performance Classification Accuracies by AI/ML Model and Training Size

The classifiers for individual performance also did well although generally not as well as the teams did. However, there was also much less variance among the classification accuracies for individual performance. The accuracies ranged from 79% at the lowest to 89% at the highest. So, while the highest accuracies weren't quite as good as they were for team performance, the lowest accuracies were a lot better than they were for team performance. Here you see each model for the most part trend upwards with a larger training set, although support vector machines with a Poly kernel did dip in accuracy quite a bit going from 60% training data to 70% training data. The best performing classifier this time was not Naïve Bayes but actually Random Forest which performed the best at every single training size.

For both team and individual performance these classifiers performed even better than expected. The data for both was split into 3 equal groups meaning randomly selecting a class for each data point should result in around 33% classification accuracy. All of the models performed more than twice as well as this in every situation. The feature selection process in this project was much more involved than previous ones using this dataset and due to that it was good to see that the classification accuracies had also largely surpassed those of the previous models created on this dataset.

Conclusion

In this study, we presented an analysis of team communication data and the measure of situation awareness within this communication data. The analysis was largely based on natural language processing with the goal of finding the best way to have a machine predict the level of situational awareness for a team or an individual. The large focus of this study was deriving the features of the data for the machine to use to make its predictions. Another goal with feature derivation was to investigate more features than the previous study on this dataset that this study is based on, “Using Big Data Analytics for Sentiment Analysis to Explore Team Communication Dynamics in Human Machine Interactions for Team Situational Awareness”. [2] That’s why this study looked into multiple types of features including part of speech tagging and TF_IDF. Also dimensionality reduction was a concept this study investigated with the use of LDA. LDA’s purpose in this paper was primarily to make each set of features uniform so they could be equally combined, however there was also a hope that this dimensionality reduction method would improve classification accuracy. This study did use mostly the same AI/ML models as the previous study on this data, just with the addition of a validation method Stratified Shuffle Split.

The classification accuracies displayed by this study improved on the accuracies in the previous study on this data and showed these high prediction accuracies could extend to evaluating individual and not just team performance. Overall, this study showed that a deeper dive into natural language processing for communication data can improve a machines ability to determine aspects of communication such as situational awareness. For future study in this area, there needs to be more investigation done into data transformation and dimensionality reduction methods such as LDA to see exactly how they improve or worsen classification capabilities, and when the best time to use them is. Furthermore, a machines ability to score messages for aspects such as their situational awareness needs to be investigated. How well can a machine generate its own scores? These are the type of questions that still need answers.

References

- [1] S. L. Marlow, C. N. Lacerenza, J. Paoletti, C. S. Burke, and E. Salas, “Does team communication represent a one-size-fits-all approach?: A meta-analysis of team communication and performance,” *Organ. Behav. Hum. Decis. Process.*, vol. 144, pp. 145–170, Jan. 2018, doi: 10.1016/j.obhdp.2017.08.001.
- [2] V. Menon, K. Weger, B. Mesmer, and S. Gholston, “Using Big Data Analytics for Sentiment Analysis to Explore Team Communication Dynamics in Human Machine Interactions for Team Situational Awareness,” in *2022 IEEE 3rd International Conference on Human-Machine Systems (ICHMS)*, Nov. 2022, pp. 1–6. doi: 10.1109/ICHMS56717.2022.9980604.
- [3] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed. O’Reilly Media, Inc., 2009.
- [4] F. Karabiber, “TF-IDF — Term Frequency-Inverse Document Frequency,” *learndatasci*. <https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/> (accessed Apr. 24, 2023).
- [5] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [6] C. Hutto and E. Gilbert, “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text,” *Proc. Int. AAAI Conf. Web Soc. Media*, vol. 8, no. 1, Art. no. 1, May 2014, doi: 10.1609/icwsm.v8i1.14550.
- [7] *Understanding support vector machines*, 1st ed. O’Reilly Media, Inc., 2017. Accessed: Apr. 24, 2023. [Online]. Available: <https://learning.oreilly.com/library/view/understanding-support-vector/9781491978733/ch01.html>
- [8] J. Chen, H. Huang, S. Tian, and Y. Qu, “Feature selection for text classification with Naïve Bayes,” *Expert Syst. Appl.*, vol. 36, no. 3, Part 1, pp. 5432–5435, Apr. 2009, doi: 10.1016/j.eswa.2008.06.054.
- [9] T. Singla, V. Gaur, and D. K. Misra, “Comparison between Multinomial Naive Bayes and Multi-Layer Perceptron for Product Review In Real Time,” in *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, May 2022, pp. 374–379. doi: 10.1109/COM-IT-CON54601.2022.9850639.
- [10] G. Hackeling, *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2017.

Appendix

Code is located at: <https://github.com/kfletc/TeamCommunicationLanguageProcessing>

Code will also be zipped and submitted along with this paper.

A video presentation and demo will also be in the zip file with this paper

Python Files:

Main.py - main for doing classification of performance by individuals and teams based on text features, message counts and frequencies, part of speech tagging, and sentiment features

GetFeatures.py - file for doing tf_idf or bag of words for categorized data and also for extracting features from categorized data

RunModels.py - file to run classification algorithms given the features and categories. Performs Stratified Shuffle Split on the data before running it through the classifiers.

CategorizeData.py - for creating cvs and pickle files with teams and individuals classified into performance categories based on mode and average of SA scores. Also preprocesses text and other derived features such as message frequency, part of speech tagging, and sentiment analysis. Groups all text data with all of these features

CleanDataset.py - for removing metadata, naming columns, and splitting up multiple SA scores