# Natural Language Predictive Model for Categorizing Tasks

*Name removed*

### Air Force Institute of Technology

### *Abstract* — Official taskings and request for information within headquarters Air Force are disseminated through the Task Management Tool (TMT) application. Currently, assigning these tasks is an arduous, time consuming manual process. This study assesses the feasibility of replacing this manual process with Machine Learning informed predictions. Focused on a single Headquarters Directorate, the study assesses if the model could accurately predict the correct division a task should be assigned to. Two models were developed, classical and neural network, and were trained on tasks received between 2015 and 2020 (over 2000 data points). Using natural language analysis techniques, the models ingested tens of thousands of words to generate a prediction. The classical approach achieved a prediction accuracy of approximately 60% while the neural network model achieved an accuracy of approximately 80%. Both models show improvements over random guessing (33%) or always tasking the most commonly tasked division (~50%). Given the limited time span and resources assigned to this study, sufficiently good results were achieved which leads the author to believe further study and investments in this area could lead to significant process improvement and resource savings for the Department of the Air Force.

*Keywords*  — Machine Learning, Neural Network, Classification, Multiclassification, Data Analytics, Predictive Model, Natural Language, Keras, NLTK

### I. Introduction, Background, & Business understanding

Within the Pentagon, all official requests for information, coordination of documents, or other formal tasks occur within the Task Management Tool (TMT) system. When an office is assigned a task, a TMT manager with knowledge of the organization must read through the task and determine which team is primarily responsible for providing a response. This activity, reading and assigning tasks, could likely be automated using Machine Learning and save the organization time and manpower.

This study will focus specifically on tasks coming into SAF/AQR, the Deputy Assistant Secretary of the Air Force for Science, Technology, and Engineering. This Directorate is responsible for oversight, policy, and strategy for all Air Force acquisition activities related to Science, Technology, and Engineering. This Directorate is made of three divisions: SAF/AQRE (responsible for engineering policy and oversight), SAF/AQRM (responsible for funding, media, and congressional engagements), and SAF/AQRT (responsible for science and technology initiatives). This Directorate is comprised of approximately 50 individuals and is regularly tasked to provide information for senior leader review, coordinate on strategy, review and develop policy, oversee technical aspects of acquisition programs, and more.

As this Directorate is tasked, it takes significant time to review incoming tasks and assign to the appropriate division. To further compound this problem, a task won’t get assigned until the TMT manager has time to review and assign the task. On some occasions, tasks can fail to be correctly assigned for multiple days due to unavailability of the TMT manager or incorrect tasking, such as tasking it to the wrong division. If this task review and assigning process could be reliably automated using Machine Learning, it could save considerable time and energy across the Pentagon and directly support the National Defense Strategy’s third line of effort: “Reform the Department’s business practices for greater performance and affordability” [1].

Consequently, the research question addressed in this study is: using available data, how well can classical machine learning and neural network algorithms predict task assignments within an Air Force headquarters directorate?

This initial effort will be considered successful if the neural network and/or classical machine learning algorithm can discern appropriate task assignments at an appreciably higher rate that “random” assigning. For this study, “random” assigning should be accurate only 33% percent of the time (one in three chance of assigning to the right Division). Therefore an initial target goal of ~60% accuracy will be set. This accuracy level should prove the concept while leaving sufficient room for future model refinement.

1. *Data Acquisition*

TMT retains archived records of all completed tasks. The TMT developer also allows users to create “custom views” that enable them to review these archived tasks based on specific user inputs [2]. For the purpose of this paper, all SAF/AQR completed tasks from the years of 2015-2020 were included. TMT also comes equipped with an export feature that allows high level information about these tasks to be exported to excel. This information was then saved in .csv format to enable easier python ingestion.

1. *Data Understanding*

The data used in this study has four primary labels: Task ID, Subject, Category, and Assigned Team. Task ID is an alphanumeric code that is unique to every TMT task. Subject is the title for the task as articulated by the office that originally drafted the task; it is typically narrative text, but numbers and special characters are allowed. *Category* is the type of task being described. *Type of task* is selectable by the task generator and includes things such as “Congressional”, “OSD Task”, “AF publication”, or simply and generically “task”. Finally, *assigned team* is the Division responsible for completing the task. All other unnecessary data (dates and times, task priority, etc) were deleted from the .csv file early in the data understanding phase.

To start, the data from the .csv file was imported to a pandas DataFrame and the “Task ID” column was removed. Looking at the data, it was immediately clear that uninformative special characters, unpredictable capitalization, and other inconsistencies needed to be fixed. Using built in python functions, all special characters were removed and all letters were forced to lowercase. In order to further handle and interpret natural language, the Natural Language Toolkit (NLTK) was used. During the model building phase, NLTK allowed this study to remove valueless words (like “for”, “a”, and “and”), tokenize/quantify the words, and more [3].

At this point, the data is sufficiently usable and can be visualized. One of the first things to do for any classification problem is to assess how balanced the data is. This allowed the study team to see if the classes are approximately evenly represented [4]. As depicted in figure 1, each of the three SAF/AQR divisions are reasonably well represented in the directorate taskings. Although AQRE owns approximately half the Directorate’s tasks, the other two divisions own a sufficient number of tasks to still consider this a balanced dataset. Table 1 shows the exact number of tasks assigned to each division during the selected timespan.

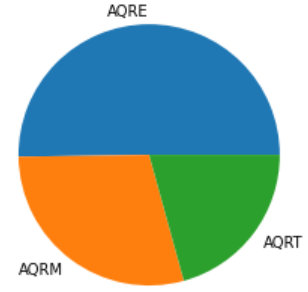


Fig. 1. Completed SAF/AQR tasks, by division

Table 1. Tasks Completed by SAF/AQR Divisions

|  |  |
| --- | --- |
| Division | Completed Tasks |
| AQRE  AQRM | 1064  616 |
| AQRT | 439 |

Next, the “Category” of task was vsiualized. As shown in Figure 2, the vast majority of completed tasks are simply “tasks” with a smaller amount of tasks being labeled as similarly generic “package”, “OSD (task)”, and “AF publication”. Because most of these categories are generic and not unique to any of the AQR divisions, is it possible this information will be of limited value.

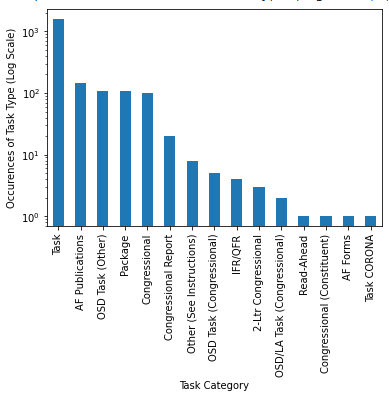


Fig. 2. Quantity of SAF/AQR task Categories (Log scale)

Another key concept in natural language processing is tokenization. Tokenizing words is a process of mapping all words to different identifiers in a large array [5]. Because these individual words will eventually be used to build our task assigning model, it is useful to know which words appear more frequently. Table 2 shows the most frequently appearing 50 words, along with their number of occurrences.

Table 2. Top 50 most frequent words in SAF/AQR tasks

|  |  |  |  |
| --- | --- | --- | --- |
| Word | Frequency | Word | Frequency |
| Air  Force | 289  276 | Draft  Update | 92  87 |
| Coordination | 249 | Meeting | 85 |
| Ltr | 213 | Pairs | 78 |
| Afi | 203 | Board | 78 |
| Fy | 190 | System | 78 |
| Coord | 189 | Majcom | 77 |
| Acquisition | 173 | Strategy | 76 |
| The | 160 | AFPD | 75 |
| Program | 154 | Technology | 74 |
| Review | 152 | Af | 73 |
| Letter | 129 | Data | 72 |
| Task | 127 | Development | 70 |
| Management | 123 | RFI | 67 |
| Aq | 118 | B | 66 |
| Report | 114 | Secaf | 63 |
| Request | 111 | Rewrite | 61 |
| Haf | 111 | Policy | 60 |
| Formal | 108 | Systems | 59 |
| Gao | 105 | DoDI | 59 |
| Dod | 104 | Research | 58 |
| Saf | 104 | PEO | 58 |
| Call | 95 | Action | 57 |
| Plan | 93 | Programs | 55 |
| Defense | 93 | ADM | 54 |

At first glance, the most frequent words do not appear immediately useful due to the preponderance of common words that are not unique to any of the teams. Words like “air”, “force”, “call”, “SAF”, “AQ”, “SECAF”, etc will likely not yield any appreciable predictability. However, many of the terms are actually quite unique to certain teams. For example, because the AQRE division has an embedded policy team that works on behalf of all of SAF/AQR, terms like “AFI” (Air Force Instruction), “AFPD” (Air Force Policy Directive), and “policy” are likely unique terms to AQRE. Similarly, as the lead for Science and Technology initiatives, terms like “technology” and “research” may be equally applicable to AQRT. Finally, as the lead to funding and strategic outreach, terms like “GAO” (Government Accountability Office) and “FY” (Fiscal Year) likely apply more to AQRM. Finally, it is clear that some words, such as “the” and “B” are highly occurring words that are simply too generic to add any value. As mentioned earlier, the NLTK toolkit will be employed during the model building phase to further massage the data and remove these “stop words”, a critical step in effective natural language processing [6].

### II. Method

The following sections will describe the data preparation activities as well as the classical and neural network model development. The data preparation and classical modeling efforts continued to rely heavily on the NLTK python products while the neural network efforts leveraged the Keras python package.

1. *Data Preparation*

The most significant activity in the data preparation phase is tokenization, or mapping the most commonly used words into a large array of ones and zeros. In this array an integer of “1” signifies the word was used in the subject of a given task while a “0” signifies it was not. This numerical representation of the words represented in a task’s subject can then be used to numerically predict the assigned team. A built in tokenization function in the NLTK package was used for this action [7]. For some of the classical modeling variations discussed later, a stemming function was used prior to being tokenized into an array. Stemming is the process of reducing similar words down to their root word. For example, run, running, runner, and runners would all be reduced to simply the word “run”. By doing this, stemming has the potential to improve predictive accuracy by allowing the model to make connections across closely dissimilar words [8].

For the classical modeling effort, no significant modification are required for the predicted variable, assigned team. For the neural network modelling effort, however, a similar tokenization action is required. In this instance, there are only three possible options for the assigned team. Therefore, the neural network simply requires an array with three columns and a row for each task. In this array, a one represents the task was assigned to the team that column represents, while a zero means it was not.

1. *Classical Modeling*

Four different classical modeling variations were developed. First, the array generated from the tokenization function was used to predict the assigned team. Second, the task category variable was added to the array to determine if a including the task category as an additional word (or words) had an appreciable difference. Third, the words were stemmed prior to being tokenized and used to predict assigned team. Finally, the task category was added to the array *and* the words were stemmed.

The primary thing to note, was the neither stemming, nor adding in the task category, had an appreciable difference on the model’s predictive accuracy. In fact, all four variations fell within an accuracy range of approximately five percentage points (~59-62% accuracy).

The main way this model could be overfit would be including *too many* words in the array doing the predicting. To avoid this, an algorithm was developed to assess when adding words to the array no longer appreciably increased the accuracy of the model. This was done by building many models and assessing the accuracy of each. The models were based on varying predictor array sizes, ranging from only one word being used (i,e, the most common word across all subjects) to a high quantity of words such that no accuracy improvements were made with additional words (1500 in the case of this experiment). Figure 3 shows how the number of words used affects the accuracy of the model. Although an initial peak occurs around 200 words, the model accuracy drops back down before slowly improving with additional words. As the figure shows, the model stops improving accuracy after about 1000 words. Therefore, to avoid overfitting, the classical modeling effort here uses the most common 1000 words as predictors.

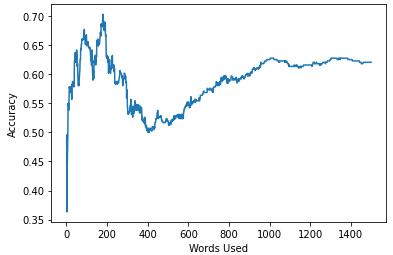


Fig. 3. Classical Model Accuracy vs Number of Predictor Words

1. *Neural Network Modeling*

Because this is a multi-classification model, the most appropriate loss function is categorical cross-entropy. This is the default loss function for multi-classification problems and is mathematically effective when the predicted label is represented by integers, as it is within the Keras package [9]. The Neural Network being used in fairly complex, with multiple layers, multiple nodes in each layers, and three output nodes. Because this neural network has a decent level of complexity, the “adam” optimizer was used. Adam is generally considered to be a popular and effective optimizer for models with some level of complexity because it carries the advantages of momentum and adaptive learning rates than allow for rapid convergence on a solution [10].

To prevent overfitting, a loss plot showing training error and testing error was assessed. As shown in figure 4, when using 20 epochs, the loss between the training and test errors is just beginning to diverge. However, as additional epochs are added (60 epochs in fig 5), the loss is clearly diverging which is indicative of an overfit model. In general, it is preferable to limit model complexity to the point when the test error begins to rise and the test and training errors are diverging [11]. As seen in figure 5, this occurs at approximately 20-30 epochs.

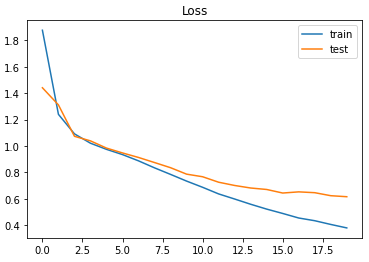


Fig. 4. Test and Training Loss vs Number of Epochs (20 Epochs)

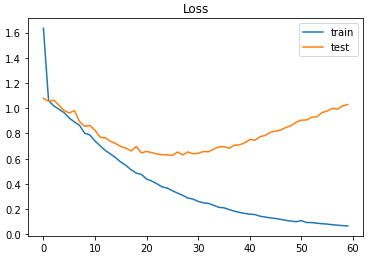


Fig. 5. Test and Training Loss vs Number of Epochs (60 Epochs)

When developing the neural network model, sweeping algorithms were developed for both batch size, the number of epochs, and number of layers. Nested loops were used to build models and assess their accuracy across batch size, number of epochs, and number of layers to discover somewhat optimal values to build a neural network with. For this problem, a batch size of 200, 25 epochs, and 2 layer was deemed optimal with an accuracy of approximately 78%.

### III. Analysis & Results

The following sections will describe the analysis and results of the classical and neural network models. The results and discussions below will include metrics for all classical model variations as well as hyper parameter adjustments for the neural network.

1. *Classical Modeling*

Metrics to assess the predictiveness and quality of the models can be seen in Table 3 below. As shown in the table, there is very little difference between the accuracy, balanced accuracy, precision, or recall across the four model variations. Choosing the “best” model across these four variations is difficult because there is so little difference. However, the second variation in the table could be considered better than the rest due to it having a *slightly* higher accuracy without resorting to more complex techniques like stemming.

Of note in this table, the precision for AQRT predicted tasks is noticeably lower than the other two divisions. This could be a potential future correction/improvement to the model.

Table 3. Classical Modelling of Task Predictions

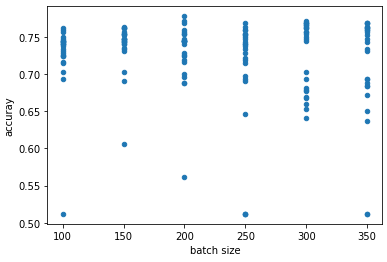
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | accuracy | Balance  accuracy | Precision | Recall |
| 1000 predictor words, unaltered | 0.60 | 0.60 | AQRE: 0.79  AQRM: 0.61  AQRT: 0.37 | AQRE: 0.58  AQRM:0.62  AQRT: 0.61 |
| 1000 predictor words, with “Category” included | 0.61 | 0.61 | AQRE: 0.79  AQRM: 0.63  AQRT: 0.38 | AQRE: 0.57  AQRM:0.66  AQRT: 0.60 |
| 1000 predictor words, with stemming | 0.60 | 0.60 | AQRE: 0.79  AQRM: 0.61  AQRT: 0.37 | AQRE: 0.58  AQRM:0.62  AQRT: 0.61 |
| 1000 predictor words, with “Category” included, with stemming | 0.61 | 0.61 | AQRE: 0.79  AQRM: 0.63  AQRT: 0.38 | AQRE: 0.57  AQRM:0.66  AQRT: 0.60 |
| Chance | 0.33 | -- | -- | -- |
| Always Predicts AQRE | 0.50 | -- | -- | -- |
| Goal | 0.60 | -- | -- | -- |

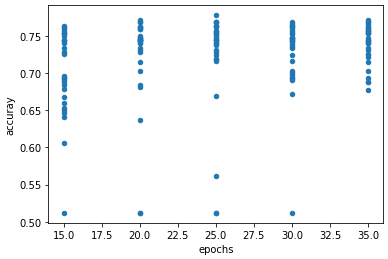
1. *Neural Network Modeling*

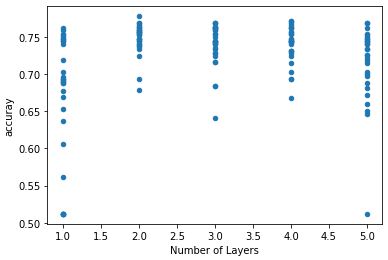
As explained above, sweeps were performed on batch size, number of epochs, and number of layers. Table 4 below shows the trade space in which sweeps were performed. Figures 6-8 show how these sweeps affected the accuracy prediction of the neural network. A few conclusions can be drawn from the figures. First, it appears a second layer very much improves the prediction, but further added layers has little to no effect (per figure 8). Second, 25 epochs seems to have the densest collection of points at higher accuracies, although not by much (per figure 7).

Table 4. Sweeping Trade Space in Neural Network Development

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hyper  Parameter | Value1 | Value2 | Value3 | Value4 | Value5 | Value6 |
| Batch  Size | 100 | 150 | 200 | 250 | 300 | 350 |
| Epochs | 15 | 20 | 25 | 30 | 35 |  |
| Internal Layers | 1 | 2 | 3 | 4 | 5 |  |

Fig. 6. Hyperparameter Sweep: Batch Size Effect on Accuracy

Fig. 7. Hyperparameter Sweep: Epochs Effect on Accuracy

Fig. 8. Hyperparameter Sweep: Number of Layers Effect on Accuracy

The primary method of regularization for this modelling effort was visual inspection of the loss plots shown in figure 4 and 5 above. As those figures show, using approximately 20-25 epochs should effectively prevent overfitting due to the epoch hyperparameter.

Early in the neural network development, before the above sweeps were conducted, the study team was able to somewhat quickly find a network that had approximately 65-75% accuracy. Fine tuning by using these sweeps, the team was able to somewhat optimize a model to achieve 78% percent accuracy.

1. *Model* *Evaluation*

Although difficult to discern, it can be observed from figures 6-8 that that the highest accuracy point is at a batch size of 200, 25 epochs, and 2 layers. This point produces an accuracy of approximately 78%, which well exceeds the initial goal of 60% accuracy, exceeds the classical model accuracy of approximately 61% accuracy, and far exceeds random guessing (~33%) or always assigning to the most commonly assigned team (~50%). The predictive accuracy of this neural network shows great promise and has the potential to provide significant to the Department of the Air Force, as described in later sections.

Ensuring overfitting was not an issue was an important consideration in model development. As shown in figures 4 and 5 above, approximately 20-25 epochs ensures the model is properly fit. Also, figure 8 shows that adding additional layers Past layer 2 does not improve the accuracy of the model (i.e. it would add complexity to the model and result in overfitting).

One final note for future model considerations, this modelling effort originally attempted to perform a hyperparameter sweep for number of neurons in each layer as well. However, given the large data array and the number of sweeps being conducted already, this optimization was dropped because the current model’s computational environment could not handle the computational complexity. Given sufficient time and resources, the number of neurons in each layer could be optimized as well.

1. *Model Application (a.k.a. deployment)*

Given sufficient resources and time to mature the models and train/scale them to fit other Headquarter directorates, algorithms like these could easily be used to predict accurate taskings across the air staff. If integrated with the TMT application, these algorithms could even be used to *automate* these taskings.

Automating these taskings across the air staff could enable the Air Force to reallocate significant manpower to higher priority activities. For example, the data and models presented above was specific to SAF/AQR, a single directorate. Assigning tasks within SAF/AQR accounts for about 25% of one person’s entire job. SAF/AQ has 11 other directorates. Automating tasking would therefore equate to approximately three positions that could be allocated to higher priority activities. SAF/AQ is also just one of approximately 20-25 Assistant Secretary or Deputy Chiefs of Staff, meaning the manpower savings across the air staff could be as great as 50-75 positions. Also, automating tasking would result in the tasks getting staffed to the Action Officers that actually provide the answers much more quickly, greatly reducing air staff inefficiencies.

### IV. Conclusion

This modelling effort attempted to asses, using available data, how well classical machine learning and neural network algorithms could predict task assignments within an Air Force headquarters directorate. With an initial target of 60% accuracy, this modelling effort successfully showed that the classical machine learning and neural networks could both yield somewhat reliable models; with the classical modelling effort yielding roughly 61% accuracy and the neural network model yielding approximately 78% accuracy.

This study describes the business understanding, data preparation, model development, model optimization, and results from the modelling effort. Finally, this study describes potential model improvements (such as optimizing the number of neurons) and ways in which this model could be expanded and applied in practice to yield great benefit to the Headquarters Air Force Mission.

### References

|  |  |
| --- | --- |
| [1] | J. Mattis, "Summary of the National Defense Strategy of the United States of Americe," Department of Defense, Washington D.C., 2018. |
| [2] | Accenture, *TMT - How to Create and Share a Custom View,* Dublin, Ireland, 2020. |
| [3] | "Natural Language Toolkit," NLTK Project, 13 April 2020. [Online]. Available: https://www.nltk.org/. [Accessed 6 February 2021]. |
| [4] | H. Tripathi, "What Is Balanced And Imbalanced Dataset?," Medium, 19 September 2019. [Online]. Available: https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5. [Accessed 4 February 2021]. |
| [5] | A. Géron, Hand-On Machine Learning with Scikit-Learn, Keras & Tensorflow, Sebastopol, CA: O'Riley, 2019. |
| [6] | T. B. II, Natural Language Processing With Python: Implementing Machine Learning and Deep Learning Algorithms for Natural Language Processes, San Francisco, CA: Apress, 2018. |
| [7] | "nltk.tokenize package," NLTK Project, 13 April 2020. [Online]. Available: https://www.nltk.org/api/nltk.tokenize.html. [Accessed 24 February 2021]. |
| [8] | "nltk.stem package," NLTK Project, 13 April 2020. [Online]. Available: https://www.nltk.org/api/nltk.stem.html. [Accessed 24 February 2021]. |
| [9] | J. Brownlee, "How to Choose Loss Functions When Training Deep Learning Neural Networks," Machine Learning Mastery, 25 August 2020. [Online]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/. [Accessed 21 February 2021]. |
| [10] | C. Hanson, "Optimizers Explained - Adam, Momentum and Stochastic Gradient Descent," Machine Learning from Scratch, 16 October 2019. [Online]. Available: https://mlfromscratch.com/optimizers-explained/#/. [Accessed 21 February 2021]. |