CS230

Real-Time Risk Evaluation System for Aviation Safety

Saakaar Bhatnagar(06340885), Nicolas Tragus(06358333)

Abstract

The goal of this work is to create an algorithm capable of evaluating in real-time the risk of a developing emergency situation in an aircraft and act as a decision support system by advising the best course of action based on previous similar situations.

1 Introduction

Even though planes are the safest mode of transportation, mid-air incidents and accidents do occur and are often sudden and unavoidable; and when a situation develops in flight, that is where a pilot earns his money by making quick decisions, often in the face of incomplete information. In a crisis, pilots have to decide between courses of actions that can potentially endanger lives versus courses that can cause enormous monetary losses, e.g. by diverting to the nearest airport. In light of this, the current work aims at developing a too that can quickly quantify the risk of a situation in the air, and can guide pilots to take actions in light of historical data, i.e. what pilots did before similar situations, and based on previous outcomes, perhaps suggest a best decision.

The raw data used here will consist of reports from The Aviation Safety Reporting System (ASRS), which collects reports from previous incidents in order to analyze and improve aviation safety. Each incident has an exhaustive set of data points (X) and results of the incident (Y), that can be used as training data.

We use a combination of LSTM and fully connected layers to analyze the information given as categorical strings (almost all data), and a sentiment analysis or word embedding for the narrative written by the pilots. Then, a combination of dense layers with a softmax activation outputs the probability for every risk class given the current situation.

2 Related work

The paper "Ensemble machine learning models for aviation incident risk prediction" [1] deals with the same issue as ours. It uses a hybrid of SVMs and Deep Neural Networks to quantify the risk of a wide range of hazardous events. This is one of very few papers that exists on the topic.

The narratives are written by pilots often using abbreviations common in the aviation field. This may be an issue when analysing the narratives, and the paper "Deep learning for extracting word-level meaning from safety report narratives" [2] tackles this issue.

Our work dffer from those approaches in that we explore several different architectures for risk quantification from those described above, and our classification is based on a smaller number of input features deemed more important by domain knowledge. The choice of inputs also makes it possible to use this tool in real time as opposed to [1].

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and features

Our dataset is composed of around 55,000 reports from the ASRS, and represents the events that happened between January 2005 and January 2020 (included) for passenger, personnal, or cargo/freight aircrafts. It must be noted that the ASRS only contains reports from flights in the USA. A typical report contains basic information such as time/place/day, or data about the aircraft and onboard crew, but most importantly a synopsis and a narrative. The former is a quick description of the event, while the latter is a small report written by the pilot describing what happened. An example of a report can be found in the main github depository.

We retained five entries to begin with: weather, flight type, flight phase, situation, and crew size, plus the narrative. After noticing some incoherence in the weather entries, such as numbers, we removed the weather for some predictions and noticed a slight increase in the performance of our models. We believe this to be due to bad data in the column. Based on domain knowledge, there is reason to believe that these are the features that affect the risk levels of a situation the most. However, the model can be easily adapted to a different number of entries, as the format of the ignored entries is the same as the entries used here.

The repartition of class is given in table 1. The number of reports can vary as much as 100% from one class to another, which causes training set imbalance. We will show the effect of this later and remedy by balancing out class labels.

The data can be decomposed into three types: Categorical strings, integer, and large paragraphs of text. Almost all the data is given as a string, except for the narrative (text) and the crew size (integer).

Class of risk	0	1	2	3	4
Number of reports	10709	8245	17621	8220	10813

Table 1: Number of reports in each class of risk. Note the predominance of class 2.

3.1 Preprocessing

As the crew size is already an integer, it doesn't need any preprocessing to be used as an input in the network.

For the 3 strings left out of our 4 entries, we need to extract them from the reports, and remove NaN/faulty datapoints that could occur. Then we need to preprocess them to remove punctuation and uppercase words. Next, we fit tokenizers to every input feature, and get the vocab sizes for that input. This will be needed to pass to the embedding layers in the network. Finally, due to variable input sizes for each feature, the largest input size is found for each input feature, and the rest are padded with zeros to obtain a rectangular input matrix of data. This procedure remains the same across our network architectures regardless of the method of dealing with the narrative.

The narrative data consists of large paragraphs. As will be described below, we input it to the network in several different ways, but we also carry out the process of removing punctuation and uppercase words beforehand.

The output data (Y) used for the training is the result of an event (e.g. aircraft damaged, general maintenance action, etc...). For it to be understood by the network, we map every output to a level of risk varying from 0-4 to use as labels. We then convert this to a one-hot encoded variant of the labels, to enable training using a softmax output.

4 Approach

We take three approaches to the architecture of the network. It is important to keep in mind the three types of inputs: categorical text, narrative and integer. Our 3 architectures differ mainly in the dealing of the narrative

1. We calculate a "sentiment score" using a predefined library (Textblob) for the narrative. It is calculated as *0.9*(optimism calculated) + 0.1*(subjectivity of narrative)*. This outputs a float that can later be used as an input for our network. **This architecture will henceforth be referred to as First Network.**

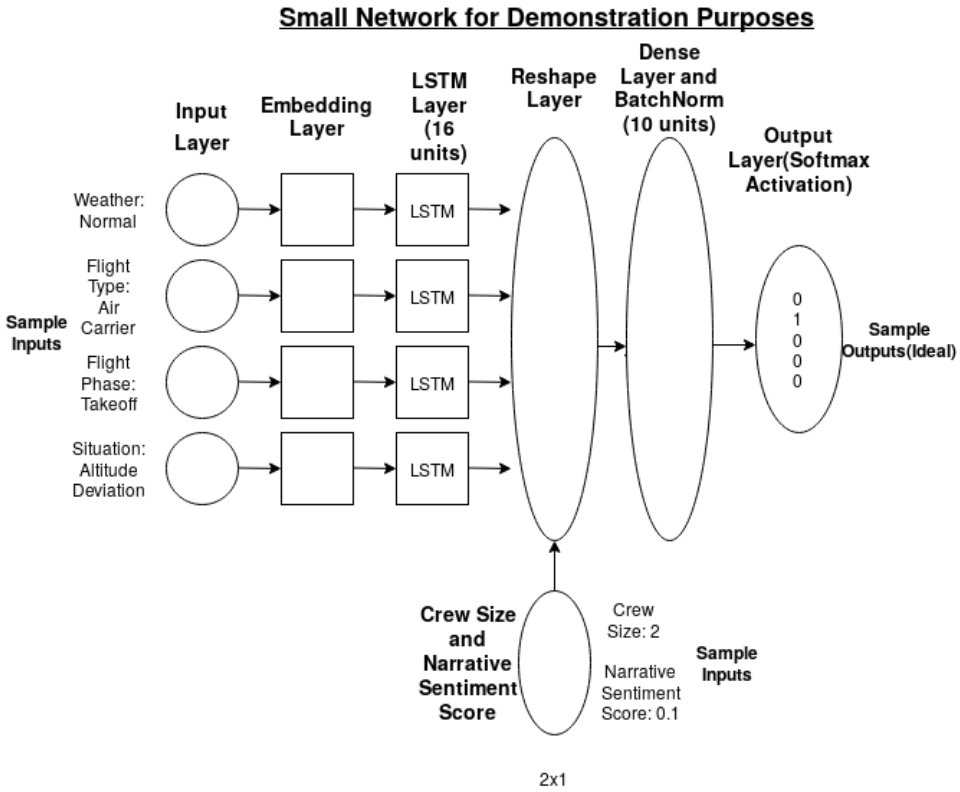


Figure 1: First network architecture. The architectures of the other two networks are substantially similar and can be found on the Git.

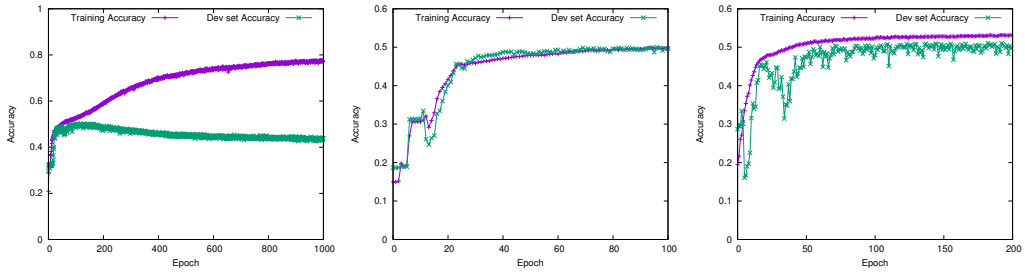
2. We learn a word embedding for the narrative data, and pass the embedded data to the LSTM layer further down the network.**This architecture will henceforth be referred to as Second Network.**
3. We use a pre-trained word embedding (GLoVE) for the embedding layer of the narrative. The embeddings for the categorical text inputs remain trainable.**This architecture will henceforth be referred to as Third Network.**

Then that data will go through LSTM layers , followed by a reshaping layer and dense layers, and finally through a softmax activation. For illustration purposes, the architecture of the first network is given in Figure 1. From the dataset, 90% is used for training and 10% for testing. As we are dealing with a mutliclass classification problem, we are using a categorical cross entropy loss.

5 Experiments, Observations and Results

Using all three architectures described above, we ran a hyper parameter search , modifying quantities like learning rate, number of layers (both dense and LSTM), regularization parameter, or number of hidden units. Some of the pertinent results such as learning curves (Figure 2) and confusion matrices (Figure 3) are shown below, along with a few important points:

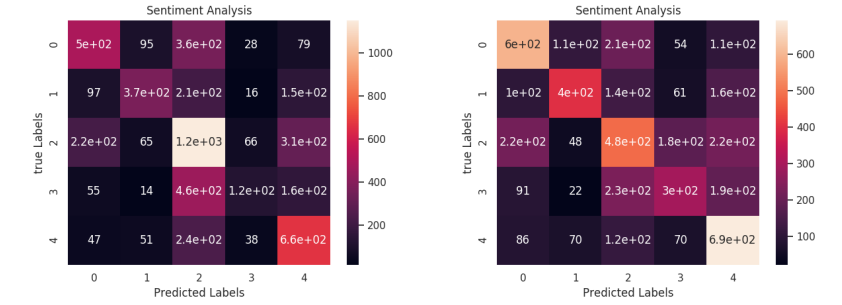
- For the first network, adding regularization did not significantly improve the overfit characteristics. The given learning curve is for $\lambda = 10^{-4}$, any higher and the network failed to smoothly learn training data. Making λ too high resulted in the network unable to learn.
- For the second and third network, tuning hyper parameters like learning rate, enabled us to go from a dev set accuracy of 0.35 to 0.5 at best, but no further as visible from the above plots.



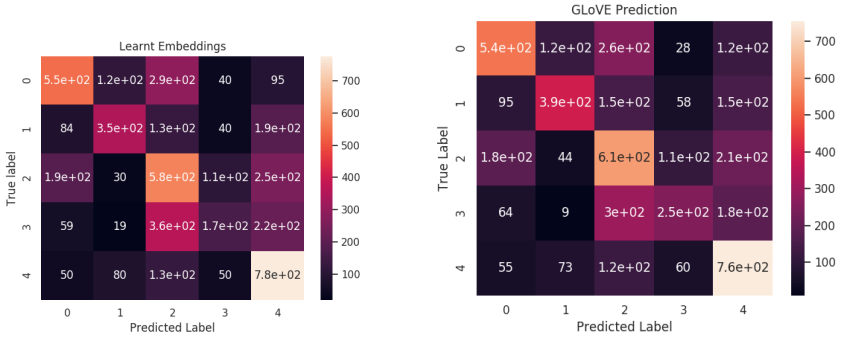
(a) First Network (b) Second Network (c) Third Network

z

Figure 2: Training curves for the three networks. In magenta, the training accuracy, and in cyan, the dev set accuracy



(a) First network before data balance (b) First network after data balance



(c) Second network after data balance (d) Third network after data balance

Figure 3: Confusion matrices for our three networks, before and after output balancing. The matrices using balanced output data are clearly more diagonal

After tuning the hyperparameters, we ran the models and got the confusion matrices for the three different architectures. We noticed the models tend to predict a middle risk level disproportionately, and hypothesized that this tendency might be due to the class imbalance present in the data. So we ran further experiments with a rebalanced dataset of size $\tilde{4}$9,000 datapoints, where we removed 6,000 points from the second class to bring it at about the same number of reports as the other classes. Here, data augmentation cannot be used given how the reports are made. In the paper from Zhang and Mahadevan [1], they approached that issue by multiplying the number of examples from other classes instead of removing reports. The resulting confusion matrices are shown in Figure 3.

Using the first network for illustration purposes, we see clearly that the networks perform better on average, over all classes, after the data is made even over output class labels. The next pertinent question is that of choosing the best algorithm out of all 3. For that, we calculate the recall for each algorithm, for each categorization. Because this is an application of safety, we want recall for the

high risk categories to be maximised, and hence the algorithm that achieves that is the best one. The formula for recall for a multi-class classification is Recall= $\frac{M_{ii}}{\sum_j M_{ij}}$

The calculation of recall for each algorithm, for each category is shown in table 2.

Risk Category	1st Net	2nd Net	3rd Net
0	0.552	0.5	0.503
1	0.467	0.441	0.462
2	0.419	0.498	0.521
3	0.355	0.200	0.311
4	0.668	0.716	0.707
Average	0.492	0.471	0.500

Table 2: Recall for each class and network architecture

From table 2 we see that if we want the network to behave best in dangerous situations, it is best to prefer the third network, as it performs better overall in the higher risk categories (3 and 4), and has a higher overall recall. Surprisingly, the first network does a better job than the second one, even though the second and same network have almost the same architecture. This may be an indicator that the GLoVE algorithm really is a better fit for our application here. We can see from the above results that although the matrices are diagonally dominant, they are not as sparse as we would like them to be (off diagonal terms are larger than we would like). We attribute this to two reasons.

First, the problem is a highly complicated one, and there might be several answers (read risk categories) to the same set of inputs (read flight conditions/events). This is reflected in the data collected from the database, and as such the network has a hard time learning the data.

Second, one should also note the human expert level error for the given problem is rather high. Even in hindsight, aviation experts tend to argue what the risk level of a situation and hence appropriate action is for a given situation. Although it is difficult to find data on this, the Bayes error can be considered to be reasonably high.

6 Conclusions and Future Work

The work has the potential to lay the groundwork for a useful system that analyzes risk in real time of a developing situation, given certain important vitals. Out of the three different architectures we tested for our model, the GLoVE model for the narrative seems to give the best performance, with a recall of 0.5. While those numbers may seem very low, one must keep in mind that even for humans, the data can be hard to classify in hindsight.

Some ways this work can be expanded are:

- If real time flight computer data like engine revolutions, or speed of ascent/descent could be added to the training data, the algorithm can become much more potent.
- The data downloaded from ASRS tends to be very "dirty", much of it and many of its columns are not usable. Better data from ASRS will enable training of a better classifier.
- One drawback of the current work is that the narratives the networks have been trained on tend to be in past tense. However, if the model was in commercial use the pilots would "speak" to the model in present tense. This has the potential to be overcome by improving the language processing part of our algorithm, perhaps by using transfer learning from another application specialized in that.

7 Contributions

Every part of the project was tackled equally by both, with Saakaar helping mostly in designing the network architectures and extracting, and Nicolas preprocessing the data and writing the report.

The entire code can be found here

References

[1] Xiaoge Zhang, Sankaran Mahadevan,"Ensemble machine learning models for aviation incident risk prediction", Decision Support Systems, Volume 116, January 2019, Pages 48-63, ISSN 0167-9236, https://doi.org/10.1016/j.dss.2018.10.009.

[2] A. Chanen, "Deep learning for extracting word-level meaning from safety report narratives," 2016 Integrated Communications Navigation and Surveillance (ICNS), Herndon, VA, 2016, pp. 5D2-1-5D2-15.