Identifying False Incidents

Brian Strickland

October 31, 2023

Abstract

The goal of this research was to build a machine learning system that could predict whether or not a false incident is being reported by a customer. By catching false incidents before they reach a lower level support tier, an organization could improve customer relations and focus support tiers on other priorities. Using term frequency matrix and Naive Bayes Classifier, a model is generated to attempt to predict if an incident is truly an incident. Using a machine learning system that is made up of data collection, data preprocessing, and feature extraction components, data is filtered down to the relevant assignment group and unnecessary features such as number, assigned to, etc are removed. The model uses a single exacted feature, False Incident, which is based on manual analysis and other criteria. If a valid prediction model can be achieve, it could later be used to automate the closing of false incidents. The research included the development of this automation via a ServiceNow business rule and Azure Function REST endpoint that would utilized the prediction model. Whenever a new incident was created within ServiceNow the REST endpoint would be called to determine if the incident could automatically be closed.

1 Introduction

1.1 Background and Problem

With over 40,000 companies using some sort of help desk management system [Enl] to track customer incidents and requests, it is important to identify false issues before they reach lower level support tiers, such as Tier 3 [Dan] where the software development team has to get involved. The reason it is important to catch these items before they reach Tier 3 is so that you can provide better customer satisfaction with reduced resolution time as well as reduce operating costs. False incidents can also have an impact on priority, meaning they could take precedence over existing work and causing delays.

1.2 System Overview

The machine learning system used to develop a model to predict whether or not an incident is truly an incident was built by collecting historic University of Central Florida (UCF) incident data that was submitted through ServiceNow. This data was then filtered down to those results only relevant to the custom solutions team. Next criteria was built to identify existing false incidents within the data and then that criteria was against the data to create a training set for the prediction model. Using an item based recommendation system, similar to what is used in Movie Recommendation System With Python and Pandas: Data Project [Dat], a vector is assigned to each known false incident to describe it. This value is then later used to compare the remaining incidents and make a prediction on whether or not they are false incidents.

1.3 Machine Learning System Components

The machine learning system components used in this project consisted of data collection, data preprocessing and feature extraction. Using this system I was able to run the following three experiments on the dataset; Cosine Similarity, Recursive Similarity and Naive Bayes Classifier.

1.3.1 Data Collection

Data was created by generating a custom service portal page within ServiceNow and placing a Data Table by Instance widget [Ser] on to the page with defined columns. Creating a custom service portal page was the only way that the more than 200K incidents were able to be exported. The data was saved as a CSV file and then loaded via a Jupyter Lab notebook. This data could had been reduced to a smaller set by using additional filters on the Data Table by Instance Widget, however I wanted to handle this within the Jupyter Lab notebook so that I could test different datasets as needed.

1.3.2 Data Preprocessing

The first step to preprocessing the data was to filter down the results to only the assignment group that we are interested in. This was done by filtering incidents to only the "Custom Application Development" group. The next step was to remove some of the unnecessary features such as assignment_group, assigned_to and number. This data is not helpful in determining anything unique to the incident because all filtered incidents are set to the same assignment group, and team members within the assignment group have no impact on whether or not the submitted incident is false. The number is just an index column for the incident record so it also doesn't provide any value in determining a false incident.
All rows that contained NaN data for any of the text content that I needed to clean was removed. This wasn't the best approached because as it removed data that could be helpful to the model. After further research, it was discovered that these rows could be handled by just replacing those values using the fillna method [Pan] which I did.
Cleaning data consisted of using a simple regular expression that removed all characters that were not letters, digits, or spaces from the the provided text.

1.3.3 Feature Extraction

There was one feature extracted from the dataset; False Incident. Records were determined to be false if they met any of the following criteria; manual analysis determined the incident was falsely reported or the incident was not related to software.

1.3.4 Experimental Results

There were three experiments carried out during this research:

- Cosine Similarity - Known false incidents were used to create a term frequency-inverse document frequency (TF-IDF). This matrix was this used to calculate similarities with all other incidents.
- Finding False Incidents Through Recursive Similarity - This experiment built off of the first experiment by looking at each incident that met the criteria to be a false incident based on the similarity score. It then used those incidents as a new list of "known false incidents" to compare against the original data set.
- Naive Bayes Classifier - finally a naive Bayes Classifier was created and was used to produce a confusion matrix and classification report.

2 Important Definitions and Problem Statement

2.1 Problem Statement

Incidents that are false should not be sent to lower level tier support teams (i.e. custom development teams). This causes impediments to work by taking away developer time and it also leads to a lower customer satisfaction rating as their requests will talk longer to resolve. However, it is hard to know whether or not an incident is false before sending it to a tier 3 support team. Also based on the amount of incidents submitted, it may not be feasible for a human to interact with every incident and classify it as a false one.

2.2 Important Definitions

- **Incident:** An ticket submitted via ServiceNow relating to an issue a user had while using a custom developed application.
- **False Incident:** An incident that is not truly an incident, but perhaps just confusion by the customer. These types of incidents should never reach lower tier support levels.
- **Custom Developed Application:** Software developed by internal Information Technology staff using formal programming languages and database systems.
- **Similarity Score:** A machine learning method that uses a nearest neighbor approach to identify the similarity of two or more objects to each other based on algorithmic distance functions. [Sim]
- **Term Frequency-Inverse Document Frequency (TF-IDF):** a numerical statistic that reflects the importance of a term in a corpus of documents.

3 Overview of Proposed System

The proposed machine learning system will be able to identify false incidents based on similarities found between text based data on an incident. This data includes text from the short description, long description and comments between support staff and the customer. The model is then trained based off of identified false incidents using feature extraction with is done through manual analysis as well as the type of incident an item is.

4 Technical Details

4.1 System Setup

The environment used to develop the machine learning pipeline consisted of a Ubuntu 22 desktop running Jupyter Lab [Jup]. Jupyter Lab was chosen as the development platform because of the data that is being used with this project. Incident records could contain confidential data to the University so options such as Kaggle or Jupyter Notebooks [Com] were not appropriate for data to be stored on. It is possible to import data over https into a notebook, but having everything self contained on a single system was also the simplest approach.

4.1.1 Azure Function and ServiceNow Business Rule

With the assumption that a valid prediction model could be achieved, an early version of the codebase was ported to an Azure Cloud function (`https://findfalseincidents.azurewebsites.net`) that when called using an HTTP GET and provided with the correct authentication token and name parameter, would return the a similarity score. A business rule was also generated within ServiceNow to automatically call this REST endpoint whenever a new incident was created within the system.

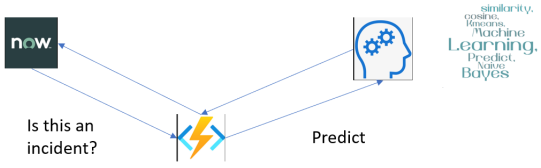

Figure 1: ServiceNow and Azure Function Prediction Model

4.2 Feature Extraction

As mentioned previously, a single feature denoting a ”False Incident” was extracted if an incident record met any two criteria.

4.2.1 Manual analysis

Manual analysis was done on the records to identify falsely reported incidents. To help identify some, an assumption that any incident that was resolved in less than an hour must be false. This filter produced a single incident record. With future investigation, it was determined that the incident was indeed false because the user stated so after being able to successfully perform a password reset. This record was then added to a data frame that would be populated with any false incident records later to be used to calculate similarity scores.

4.2.2 Type of incident

The other criteria for used to find false incident was the category type of an incident. Incidents are either categorized as hardware or software. Since all custom solutions are developed using software, any hardware incident can immediately be eliminated and used for comparison.

This feature extraction yield six incidents that could be used to train a model and generate similarity scores.

4.3 Predictive Modeling

Two types of models were used to make predictions on this data; a type based recommendation system and Naive Bayes. Both of these models were chosen for their usefulness in natural language processing.

4.3.1 Type Based Recommendation

Typed based recommendation systems recommend items to users based on similar items. This type of recommendation system is ideal to find incidents that are false since if an incident has a high similarity with a known false incident, it is likely to be false as well.

In this research cosine similarity scores are used to predict whether or not an incident is false. To do this a feature matrix based on the short description values of known incidents (i.e. FalseIncident == ”true”) is created using a term frequency-inverse document frequency (TF-IDF).

This model is then used on each incident to determine whether to determine whether or not they are false.

4.3.2 Naive Bayes

With the Naive Bayes classifier, first the most common English words from the data, stop words, were removed from the short_description field. This is done through nltk.corpus.stopwords. Feature set for the model to be trained on consisted on the short_description field. Once the data was removed of common words, it was transformed into a TF-IDF matrix format.

5 Experiments

5.1 Cosine Similarity Using TF-IDF

The cosine similarly using TF-IDF experiment is the foundation of this machine learning pipeline. By using an ngram range of 1,5 and a similarity criteria of > 0.7 the model was able to find at least two incidents that could be classified as false incidents. A range lower than 1,5 for ingram did not produce any results and anything lower than a 0.7 would not be acceptable for a predictive model. Testing different values I was able to determine I would have to set the similarity criteria to lower than 0.6 to get any additional data.

5.2 Finding False Incidents Through Recursive Similarity

To find potential similarities, lets recursively look at all nodes that have a direct similarity score of 0.7 or higher and every indirect node with a similarities score.

$$\sum_{i=1}^{n} f(child_i)$$

where $f(child_i)$ is defined as:

$$\text{if } child_i \text{ has a node with similarity score } \geq 0.7 \text{ then } return + \sum_{j=1}^{m_i} f(grandchild_{ij})$$

As a first pass, there were only two incidents that were found by comparing the initial seven known false incidents to the rest of the incident list. Using those two as a start, a search for cosine similarities of at least 0.7 was recursively done using each new set of false incidents found as a start comparison. This did produce one additional result.

Reducing the similarity restriction down to 0.5 increased the initial number of incidents found to six and found an additional seven more running the recursive algorithm. However the majority of additional findings has similarity scores close to 0.5 so that isn't very comparative [Ala]

5.3 Naive Bayes Classifier

The trained data resulted in 6 samples and 40 features and the test data resulted in 355 samples with 40 features. When using the prediction model, all predictions were made as a false prediction. (i.e. even if the incident was really an incident the model predicted it as false.).

By looking at the classification report [lea] produced from this model we can see the following:

- The precision for FalseIncidents is 0.00, meaning 100% false positive predictions.
- The recall for FalseIncidents was also 0.00, meaning 0% of them were correctly predicted.

A confusion matrix [Dha], a matrix that shows the number of true positives, true negatives, false positives and false negatives, also confirms that the prediction model is not predicting well. The confusion matrix produced 349 out of 349 instances true negatives and 6 instances of true positives.

5.4 Data Comparison

Data such as description and user comments were also interchanged with short_description to see if any better predictions could be made. However, no matter the combination of data sets the model was not very good at prediction a false incident. This is large due to the low sample set used for training data.

6 Related Work

There are a lot of topics using text to identify similar items from data sets just as this paper does. For example in "Knowledge discovery through text-based similarity searches for astronomy literature" by Wolfgang E. Kerzendorf [Ker19], used a text based recommendation system to identify astronomy articles that were similar in nature to a provided paper. The difference that my research provides is that it is looking at single specific parts of a dataset whereas Wolkfgang E. Kerzendorf was looking at an entire document for similarity.

In "Short Text Similarity with Word Embeddings" by Tom Kenter and Maarten de Rijke, they take an approach which differs from mine, but could possibly improve my predictions. Their research looks instead at the semantic, pertaining to a representation of meaning [KR15], instead of just scoring based on word likeness between two sources. Borrowing ideas from this paper in a future work could improve the overall false incident prediction model

Another related paper, "Creating a Movie Reviews Classifier Using TF-IDF in Python" by Prateek Majumder, [Maj] looked at making a movie reviews classifier using TF-IDF. This was very similar to the work I was doing. The main differences between the works was the value that was being used from the training sets. Training in my research was based on incidents labeled as a "FalseIncident" while their paper looked at a feature containing good or bad reviews.

7 Conclusions

In conclusion, the current machine learning system that was developed to predict whether or not an incident is false would not be helpful in a real world scenario as the data from the experiments show that the model is not a good predictor. However, improvements could be made to the model, mostly by providing more training data than there was available, since seven items is not enough to adequately train a model to correctly predict false incidents. Perhaps even including data from assignment groups outside of custom application development could increase the accuracy of the model.

If the prediction accuracy could be improved, the model could have the potential to improve the incident management process by identifying false incidents and preventing them from reaching lower-level support tiers such as the custom development team. Combined with the power of leveraging a REST endpoint to provide model predictions to ServiceNow, better customer satisfaction, reduced resolution time, and lower operating costs could be achieved.

8 Code

Github: https://github.com/strick/false-incident-finder/blob/main/ML%20Project.md

References

[Ala] Richmond Alake. *Understanding Cosine Similarity and Its Applications*. URL: https://builtin.com/machine-learning/cosine-similarity.

[Com] The Jupyter Community. *Introduction to the JupyterLab and Jupyter Notebooks*. URL: https://jupyter.org/try-jupyter/retro/notebooks/?path=notebooks/Intro.ipynb.

[Dan] Sophie Danby. *What Does Tier 3 Help Desk Do? Duties, Skills, and Examples*. URL: https://blog.invgate.com/tier-3-help-desk.

[Dat] Dataquest. *Movie Recommendation System With Python and Pandas: Data Project*. URL: https://www.youtube.com/watch?v=eyEabQRBMQA.

[Dha] Dharmaraj. *Understanding Confusion Matrix, Precision-Recall, and F1-Score*. URL: https://medium.com/@draj0718/understanding-confusion-matrix-precision-recall-and-f1-score-5b065f297f1b.

[Enl] Enlyft. *IT Helpdesk Management Products*. URL: https://enlyft.com/tech/it-helpdesk-management.

[Jup] Jupyter. *Installing Jupyter*. URL: https://jupyter.org/install.

[Ker19] W.E. Kerzendorf. "Knowledge discovery through text-based similarity searches for astronomy literature". In: *J Astrophys Astron* 40 (2019). URL: https://doi.org/10.1007/s12036-019-9590-5.

[KR15] Tom Kenter and Maarten de Rijke. "Short Text Similarity with Word Embeddings". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia: Association for Computing Machinery, 2015, pp. 1411–1420. ISBN: 9781450337946. DOI: 10.1145/2806416.2806475. URL: https://doi.org/10.1145/2806416.2806475.

[lea] scikit learn. *sklearn.metrics.classification*$_r$*eport*¶. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html.

[Maj] Prateek Majumder. *Creating a Movie Reviews Classifier Using TF-IDF in Python*. URL: https://www.analyticsvidhya.com/blog/2021/09/creating-a-movie-reviews-classifier-using-tf-idf-in-python/.

[Pan] Pandas. *pandas.DataFrame.fillna*. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html.

[Ser] ServiceNow. *Data table from instance definition widget*. URL: https://docs.servicenow.com/bundle/rome-servicenow-platform/page/build/service-portal/concept/data-table-widget.html.

[Sim] SimMachines. *Similarity Based Machine Learning Provides AI Transparency and Trust*. URL: https://simmachines.com/similarity-based-machine-learning-provides-ai-transparency-trust/#:~:text=Similarity%20is%20a%20machine%20learning,based%20on%20algorithmic%20distance%20functions..