Ticket Classification API Documentation

August 25, 2025

Contents

1 Overview

The Ticket Classification API is a FastAPI-based application designed to classify support tickets into predefined topic groups using a machine learning model. The API leverages a Logistic Regression model trained on TF-IDF vectorized text data, enabling accurate classification of ticket descriptions.

1.1 Purpose

This API provides an automated solution for categorizing support tickets based on their content, streamlining ticket management processes.

1.2 Key Features

- Text preprocessing to remove noise (punctuation, stopwords, etc.).
- TF-IDF vectorization with support for unigrams and bigrams.
- Logistic Regression model for multi-class classification.
- Persistent model and vectorizer loading for efficient predictions.
- RESTful endpoint for real-time ticket classification.

2 Investigation Results

The investigation into ticket classification was conducted as of August 25, 2025, analyzing a dataset of 47,837 tickets. The distribution across topic groups reveals the following: Hardware leads with 13,617 tickets (28.5%), followed by HR Support with 10,915 tickets (22.8%), Access with 7,125 tickets (14.9%), Miscellaneous with 7,060 tickets (14.8%), Storage with 2,777 tickets (5.8%), Purchase with 2,464 tickets (5.2%), Internal Project with 2,119 tickets (4.4%), and Administrative rights with 1,760 tickets (3.7%). This distribution is visually represented in a pie chart titled "Percentage of Each Topic Group," highlighting Hardware and HR Support as the most prevalent categories, together accounting for over 51% of the tickets.

3 Training

3.1 Data Preprocessing

The input text is preprocessed using the following steps to clean and normalize the data:

- Convert text to lowercase to ensure uniformity.
- Remove punctuation and special characters using regex ($[^{\backslash w \backslash s}]$).$RemoveEnglishstopwordsusingNLTK$

3.2 TF-IDF Vectorization

The preprocessed text is transformed into numerical features using a `TfidfVectorizer` with:

- `max_features=5000`
- `ngram_range=(1,2)` (unigrams and bigrams)
- English stopwords removal

3.3 Model Training

A `LogisticRegression` model is trained with:

- `random_state=42`
- `class_weight="balanced"` to handle class imbalance
- `max_iter=500` for convergence

Evaluation across the train, validation, and test sets yielded the following results:

3.3.1 Train Set Performance

	precision	recall	f1-score	support
Access	0.93	0.91	0.92	5696
Administrative rights	0.73	0.99	0.84	1407
HR Support	0.92	0.87	0.89	8726
Hardware	0.91	0.84	0.87	10890
Internal Project	0.83	0.98	0.90	1694
Miscellaneous	0.85	0.90	0.87	5646
Purchase	0.93	0.97	0.95	1957
Storage	0.87	0.97	0.92	2221
accuracy			0.89	38237
macro avg	0.87	0.93	0.90	38237
weighted avg	0.89	0.89	0.89	38237

3.3.2 Validation Set Performance

	precision	recall	f1-score	support
Access	0.90	0.88	0.89	1068
Administrative rights	0.66	0.84	0.74	264
HR Support	0.88	0.82	0.85	1637
Hardware	0.86	0.80	0.83	2042
Internal Project	0.77	0.92	0.84	318
Miscellaneous	0.80	0.86	0.83	1058
Purchase	0.89	0.95	0.92	367
Storage	0.84	0.92	0.88	416

accuracy			0.85	7170
macro avg	0.82	0.87	0.85	7170
weighted avg	0.85	0.85	0.85	7170

3.3.3 Test Set Performance

	precision	recall	f1-score	support
Access	0.91	0.88	0.89	356
Administrative rights	0.59	0.91	0.71	88
HR Support	0.89	0.85	0.87	545
Hardware	0.87	0.78	0.82	681
Internal Project	0.84	0.90	0.87	106
Miscellaneous	0.80	0.86	0.83	353
Purchase	0.88	0.93	0.90	122
Storage	0.84	0.90	0.87	139
accuracy			0.85	2390
macro avg	0.83	0.88	0.85	2390
weighted avg	0.86	0.85	0.85	2390

The model demonstrates consistent performance across all sets, with an overall accuracy of 89% on the training set, 85% on the validation set, and 85% on the test set. The macro and weighted averages indicate robust generalization, though "Administrative rights" shows lower precision, suggesting potential areas for improvement.

4 Usage

The Ticket Classification API can be utilized for real-time ticket categorization. For example, to classify a ticket, a POST request can be sent to the "/predict" endpoint, such as:

```
curl -X POST "http://localhost:8000/predict" -H "Content-Type: application/
```

This returns a response like:

```
{
    "text": "reset my account password",
    "predicted_topic": "Password Reset"
}
```

To verify the API's status, a GET request to "/" can be made:

```
curl -X GET "http://localhost:8000/"
```

Yielding:

```
{
    "message": "Ticket Classification API is running "
}
```

These examples demonstrate the API's practical application for automating ticket management based on the trained model's predictions.

5 Limitations and Future Improvements

5.1 Limitations

- Limited to English text due to NLTK stopwords.
- Model performance depends on the quality and diversity of the training data.
- Static model; requires retraining for new ticket categories.

5.2 Future Improvements

- Support for multilingual ticket classification.
- Integration of more advanced models (e.g., transformers).
- Real-time model retraining pipeline.