



## **Assignment -Solution**

### **Introduction**

The goal of this project is to develop an intelligent intent prediction model using customer feedback data from Novo, a fintech company. This model will be used to automatically label incoming customer support tickets with relevant intent categories, enabling intelligent routing and efficient handling of customer queries.

Accurate intent prediction is crucial for providing exceptional customer support. By understanding the underlying intent behind each customer's inquiry, the support team can route the ticket to the appropriate team or agent, ensuring that the issue is addressed by the most qualified personnel. This not only improves response times and customer satisfaction but also optimizes resource allocation within the support organization.

### **Understanding the Data**

The first step is to carefully examine the customer feedback data provided by Novo. This data comes from Trustpilot, a website where customers can leave reviews and share their experiences with companies. The data includes the actual text of the reviews, along with some basic information like ratings.

To work with this data, we need to clean it up a bit. This means removing any irrelevant information like HTML tags or special characters that might be present. We also need to convert all the text to lowercase and break it down into individual words (a process called tokenization).

### **Effective Intent Taxonomy-**

In the next step Design the Effective Intent Taxonomy so To design an effective intent taxonomy for the customer feedback, it's important to consider the following factors:

**Granularity:** The taxonomy should strike a balance between being too broad and too specific. Too broad categories may fail to capture the nuances of customer intents, while too specific categories may lead to an overly complex and unwieldy taxonomy.

**Hierarchical Structure:** A hierarchical structure can help organize intents in a logical and scalable manner. Higher-level categories can encompass broader topics, while lower-level categories can capture more specific intents.

**Business Context:** The taxonomy should be tailored to the specific business context and the types of customer interactions encountered. In this case, the context is related to banking and financial services, so categories should reflect common customer intents in this domain.

**Data-driven Approach:** Analyzing the provided customer feedback can reveal patterns and common themes, which can inform the development of the intent taxonomy.

Here's an example of how we can define the intent taxonomy using Python's nested dictionaries:

```
intent_taxonomy = {
    "Account": {
        "Login/Password Issues": None,
        "Account Setup": None,
        "Account Security": None,
        "Account Closure": None
    },
    "Payments": {
        "Debit Card": {
            "Declined Transactions": {
                "Fraud": None,
                "Insufficient Funds": None
            },
            "Card Replacement": None
        },
        "Checks": {
            "Mobile Deposit": {
                "Deposit Issues": None,
                "Void Checks": None
            }
        }
    }
}
```

```
        "Check Clearing": None
    },
    "Invoicing": {
        "Invoice Generation": None,
        "Invoice Payment": {
            "Unpaid": {
                "Conflict": None,
                "Pending": None
            },
            "Paid": None
        }
    }
}
```

This nested dictionary structure allows us to represent the hierarchical relationships between different intent categories and subcategories.

Alternatively, we could use a tree data structure or a more complex object-oriented approach to define the taxonomy, depending on the requirements and complexity of the problem.

## Reasoning

The reasoning behind this taxonomy is to provide a comprehensive and organized structure that can accommodate a wide range of customer intents while maintaining a logical hierarchy. By capturing intents at varying levels of granularity, the taxonomy can facilitate effective routing of customer inquiries to the appropriate support team or resource.

It's important to note that this taxonomy is a starting point and may require further refinement based on ongoing analysis of customer feedback and evolving business requirements. Continuous evaluation and adjustment of the intent taxonomy can help ensure its effectiveness in supporting an intelligent routing engine.

## Extract Intents

To automatically extract intents from the customer feedback data, we can leverage unsupervised learning techniques, such as topic modeling or clustering algorithms, in combination with natural language processing (NLP) techniques. Here's a high-level approach using Python and relevant libraries:

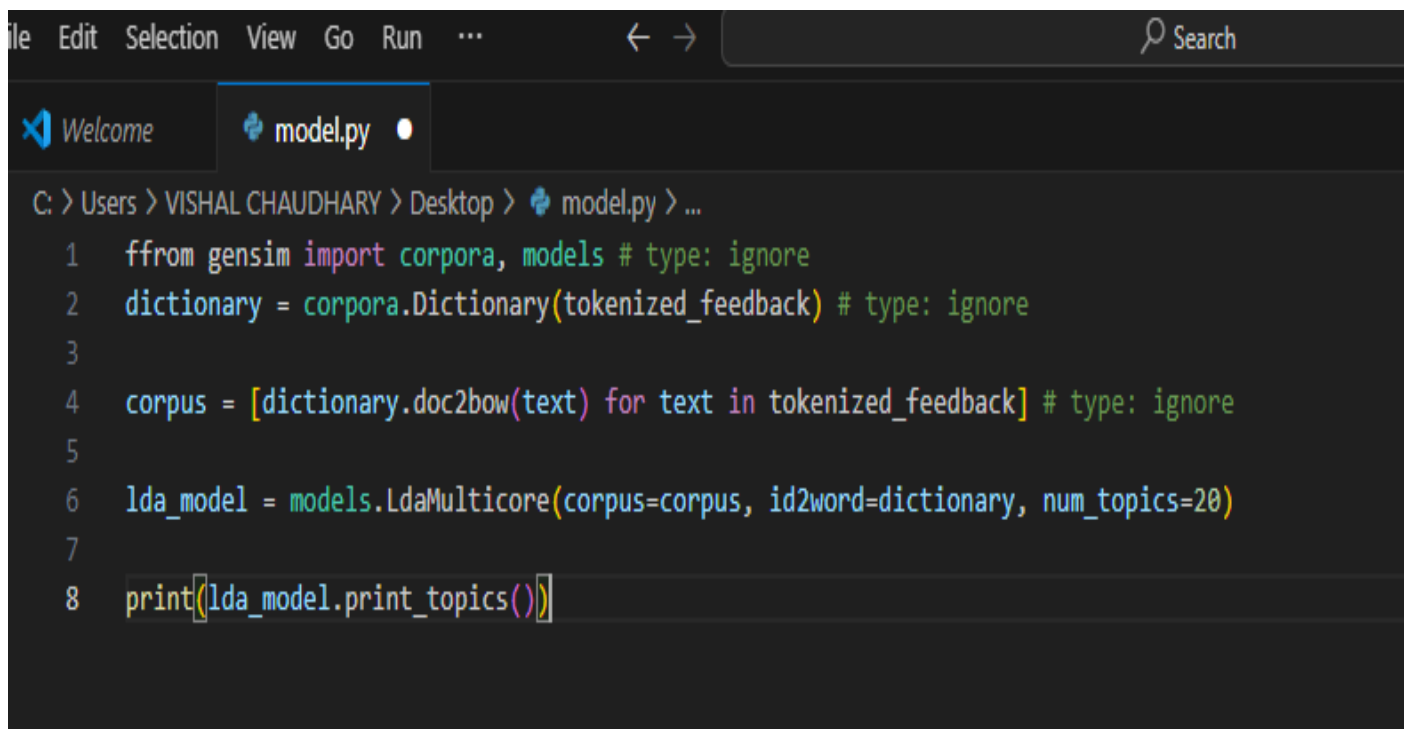
### Data Preprocessing:

- Import necessary libraries: `import pandas as pd, import re, import nltk`
- Load the customer feedback data into a Pandas DataFrame

### Topic Modeling:

- Use LDA (Latent Dirichlet Allocation) or NMF (Non-negative Matrix Factorization) for topic modeling

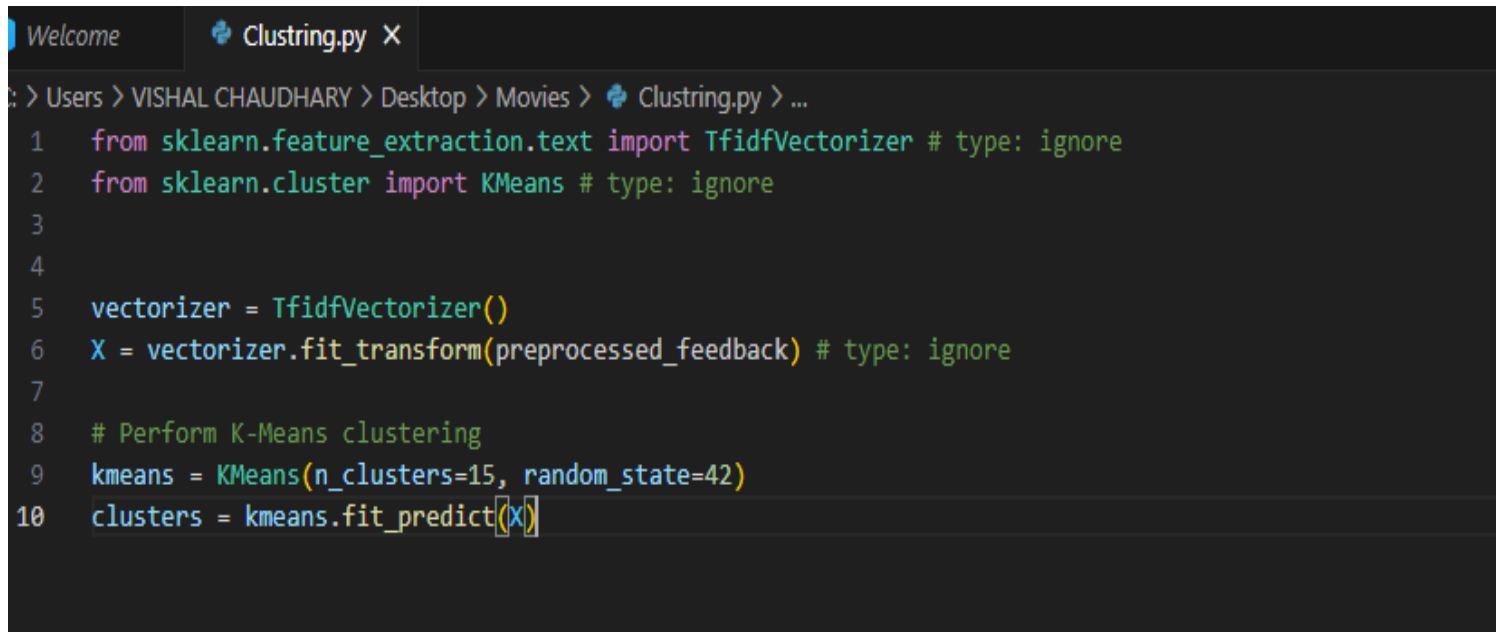
Example using gensim library for LDA:

A screenshot of a code editor window with a dark theme. The window has a menu bar at the top with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', and a search icon. Below the menu bar, there are two tabs: 'Welcome' and 'model.py'. The 'model.py' tab is active. The code is written in Python and uses the gensim library for LDA. The code is as follows:

```
C: > Users > VISHAL CHAUDHARY > Desktop > model.py > ...
1  from gensim import corpora, models # type: ignore
2  dictionary = corpora.Dictionary(tokenized_feedback) # type: ignore
3
4  corpus = [dictionary.doc2bow(text) for text in tokenized_feedback] # type: ignore
5
6  lda_model = models.LdaMulticore(corpus=corpus, id2word=dictionary, num_topics=20)
7
8  print(lda_model.print_topics())
```

## Clustering:

- Use K-Means clustering or hierarchical clustering algorithms
- Example using `sklearn` library for K-Means:



```
Welcome Clustering.py X
: > Users > VISHAL CHAUDHARY > Desktop > Movies > Clustering.py > ...
1 from sklearn.feature_extraction.text import TfidfVectorizer # type: ignore
2 from sklearn.cluster import KMeans # type: ignore
3
4
5 vectorizer = TfidfVectorizer()
6 X = vectorizer.fit_transform(preprocessed_feedback) # type: ignore
7
8 # Perform K-Means clustering
9 kmeans = KMeans(n_clusters=15, random_state=42)
10 clusters = kmeans.fit_predict(X)
```

## Intent Labelling:

- Manually inspect the topics or clusters and assign meaningful intent labels
- Alternatively, use techniques like topic labelling or cluster labelling

## Evaluation:

- Use metrics like silhouette score, calinski-harabasz score, or manual evaluation to assess the quality of the clusters/topics
- Iterate and refine the number of topics/clusters and preprocessing steps as needed

## Integration:

- Map the extracted intents to the intent taxonomy
- Incorporate the extracted intents into the training data for the intent prediction model

# Classify The Tickets

To teach the model to classify customer tickets according to the defined intent taxonomy, we can follow a supervised learning approach using deep learning techniques. Here's a high-level outline of the process:

## Data Preparation:

- Split the customer feedback data into training and validation sets.
- Manually label a subset of the customer feedback data with the corresponding intents from the defined taxonomy.

## Text Preprocessing:

- Convert text to lowercase
- Remove punctuation, stopwords, and special characters
- Tokenize the text into words or subword units
- Perform stemming or lemmatization

## Feature Representation:

- Use techniques like Bag-of-Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), or pre-trained word embeddings (e.g., Word2Vec, GloVe) to represent the text data numerically.

## Model Architecture:

- Choose a deep learning architecture suitable for text classification, such as Recurrent Neural Networks (RNNs) like Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), or Transformer-based models like BERT or RoBERTa.

Example using PyTorch and LSTM:

```
import torch
import torch.nn as nn

class IntentClassifier(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, output_size):
        super(IntentClassifier, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_size)

    def forward(self, x):
        embedded = self.embedding(x)
        output, _ = self.lstm(embedded)
        output = self.fc(output[:, -1, :])
        return output
```

- Define the loss function (e.g., Cross-Entropy Loss for multi-class classification)
- Choose an optimizer (e.g., Adam, SGD) and learning rate
- Train the model on the labeled training data using techniques like mini-batch gradient descent
- Monitor performance on the validation set and apply techniques like early stopping or learning rate scheduling

### **Model Evaluation:**

- Evaluate the model's performance on a held-out test set using metrics like accuracy, precision, recall, and F1-score.
- Analyze errors and misclassifications to identify areas for improvement.

### **Model Deployment:**

- Once satisfied with the model's performance, deploy it to classify new customer tickets in real-time or batch mode.

### **Continuous Learning:**

- As new customer tickets arrive, periodically retrain the model with the latest labelled data to adapt to changing patterns and emerging intents.

➤ To handle new intents emerging from the launch of a new product line, such as a payroll product, we can employ techniques like continuous learning and transfer learning to adapt the existing intent classification model. Here's a high-level approach:

### **Data Collection and Labeling:**

- Continuously collect customer feedback, reviews, and support tickets related to the new payroll product.
- Manually label a subset of this data with the appropriate intents, including any new intents specific to the payroll product.

### **Data Augmentation:**

- Leverage techniques like back-translation, synonym replacement, or data augmentation libraries (e.g., nlpaug) to create additional synthetic training data for the new intents.
- This can help increase the diversity and robustness of the training data, especially when dealing with a limited amount of real-world data for the new intents.

### **Transfer Learning:**

- Utilize the pre-trained weights from the existing intent classification model as a starting point.
- Fine-tune the model on the combined dataset, including the original labeled data and the newly labeled data for the payroll product intents.
- This approach allows the model to leverage its existing knowledge while adapting to the new domain and intents.

### **Continual Learning:**

- Employ techniques like Elastic Weight Consolidation (EWC) or Learning without Forgetting (LwF) to enable the model to learn the new intents without catastrophically forgetting the previously learned intents.
- These techniques introduce regularization terms in the loss function to preserve the important weights from the initial training.

### **Incremental Learning:**

- Instead of retraining the entire model from scratch, incrementally update the model as new labeled data becomes available.



- Use techniques like reservoir sampling or prioritized experience replay to maintain a balanced and representative buffer of training samples from both old and new intents.
- Periodically retrain the model on this buffer to incorporate the new knowledge while retaining the previously learned intents.

### **Active Learning:**

- Implement an active learning strategy to identify and prioritize the most informative unlabeled samples for manual annotation.
- Use techniques like uncertainty sampling, query-by-committee, or expected gradient length to select the samples where the model is most uncertain or likely to benefit the most from additional labeled data.
- This can help optimize the manual labeling effort and accelerate the learning process for the new intents.

### **Evaluation and Monitoring:**

- Continuously evaluate the model's performance on a held-out test set, including samples from both old and new intents.
- Monitor metrics like accuracy, precision, recall, and F1-score, separately for the new intents and overall.
- Analyze misclassifications and error patterns to identify areas for improvement or additional data collection and labeling.

### **Deployment and Updates:**

- Deploy the updated model to production, ensuring a seamless transition and minimal disruption to the existing system.

- Establish a regular cadence for retraining and updating the model as more data becomes available, striking a balance between incorporating new knowledge and maintaining stability.

By implementing these techniques, the intent classification model can effectively adapt to new product lines, emerging intents, and changing customer needs over time

## Conclusion

designing an effective intent taxonomy and developing a robust intent classification model is a crucial step in building an intelligent routing engine for customer support. By following a data-driven approach and leveraging advanced natural language processing and machine learning techniques, we can create a system that accurately captures and routes customer inquiries to the appropriate support channels.

The proposed intent taxonomy strikes a balance between granularity and simplicity, ensuring that it can accommodate a wide range of customer intents while maintaining a logical hierarchy. The taxonomy is tailored to the banking and financial services domain, reflecting common customer concerns and inquiries in this sector.

To teach the model to classify customer tickets according to the defined intent taxonomy, we can employ supervised learning techniques using deep learning architectures like Recurrent Neural Networks (RNNs) or Transformer-based models. By leveraging pre-trained language models and fine-tuning them on labelled data, we can achieve high performance and faster convergence. However, as new product lines are introduced and customer needs evolve, the intent classification model must be able to adapt and learn new intents continuously. Techniques like transfer learning, continual learning, incremental learning, and active learning can be employed to seamlessly incorporate new intents into the model without forgetting previously learned knowledge. Regular evaluation, monitoring, and analysis of model performance and error patterns are crucial to identify areas for improvement and inform the data collection and labelling efforts. By continuously updating the model

with the latest labelled data and refining the intent taxonomy as needed, we can ensure that the intelligent routing system remains accurate and relevant over time. Ultimately, the success of this endeavor lies in the careful integration of domain knowledge, cutting-edge machine learning techniques, and a commitment to continuous improvement. By leveraging the power of natural language processing and machine learning, we can deliver an exceptional customer experience and streamline support operations, positioning Novo as a leader in the banking and financial services industry.