In [11]:
```
!pip install pandas numpy transformers scikit-learn matplotlib seaborn gensim wordc
```

```
Requirement already satisfied: pandas in c:\users\ibrah\anaconda3\lib\site-package
s (1.5.3)
Requirement already satisfied: numpy in c:\users\ibrah\anaconda3\lib\site-packages
(1.24.4)
Requirement already satisfied: transformers in c:\users\ibrah\anaconda3\lib\site-p
ackages (4.37.1)
Requirement already satisfied: scikit-learn in c:\users\ibrah\anaconda3\lib\site-p
ackages (1.3.0)
Requirement already satisfied: matplotlib in c:\users\ibrah\anaconda3\lib\site-pac
kages (3.7.2)
Requirement already satisfied: seaborn in c:\users\ibrah\anaconda3\lib\site-packag
es (0.12.2)
Requirement already satisfied: gensim in c:\users\ibrah\anaconda3\lib\site-package
s (4.3.0)
Requirement already satisfied: wordcloud in c:\users\ibrah\anaconda3\lib\site-pack
ages (1.9.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\ibrah\anaconda3
\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\ibrah\anaconda3\lib\site-p
ackages (from pandas) (2023.3.post1)
Requirement already satisfied: filelock in c:\users\ibrah\anaconda3\lib\site-packa
ges (from transformers) (3.9.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in c:\users\ibrah\anac
onda3\lib\site-packages (from transformers) (0.20.3)
Requirement already satisfied: packaging>=20.0 in c:\users\ibrah\anaconda3\lib\sit
e-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in c:\users\ibrah\anaconda3\lib\site-pa
ckages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in c:\users\ibrah\anaconda3\lib\s
ite-packages (from transformers) (2022.7.9)
Requirement already satisfied: requests in c:\users\ibrah\anaconda3\lib\site-packa
ges (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in c:\users\ibrah\anaconda3
\lib\site-packages (from transformers) (0.15.1)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\ibrah\anaconda3\lib
\site-packages (from transformers) (0.3.2)
Requirement already satisfied: tqdm>=4.27 in c:\users\ibrah\anaconda3\lib\site-pac
kages (from transformers) (4.65.0)
Requirement already satisfied: scipy>=1.5.0 in c:\users\ibrah\anaconda3\lib\site-p
ackages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\ibrah\anaconda3\lib\site-
packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ibrah\anaconda3\li
b\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\ibrah\anaconda3\lib\si
te-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\ibrah\anaconda3\lib\site-p
ackages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\ibrah\anaconda3\lib\s
ite-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ibrah\anaconda3\lib\s
ite-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\ibrah\anaconda3\lib\site-
packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\ibrah\anaconda3\l
ib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\ibrah\anaconda3\lib\s
ite-packages (from gensim) (5.2.1)
Requirement already satisfied: FuzzyTM>=0.4.0 in c:\users\ibrah\anaconda3\lib\site
-packages (from gensim) (2.0.9)
Requirement already satisfied: pyfume in c:\users\ibrah\anaconda3\lib\site-package
s (from FuzzyTM>=0.4.0->gensim) (0.3.4)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\ibrah\anaconda3\lib\si
te-packages (from huggingface-hub<1.0,>=0.19.3->transformers) (2023.12.2)
```

```
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\ibrah\anacon
da3\lib\site-packages (from huggingface-hub<1.0,>=0.19.3->transformers) (4.9.0)
Requirement already satisfied: six>=1.5 in c:\users\ibrah\anaconda3\lib\site-packa
ges (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: colorama in c:\users\ibrah\anaconda3\lib\site-packa
ges (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ibrah\anaconda
3\lib\site-packages (from requests->transformers) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ibrah\anaconda3\lib\site-p
ackages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ibrah\anaconda3\lib
\site-packages (from requests->transformers) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ibrah\anaconda3\lib
\site-packages (from requests->transformers) (2023.7.22)
Requirement already satisfied: simpful==2.12.0 in c:\users\ibrah\anaconda3\lib\sit
e-packages (from pyfume->FuzzyTM>=0.4.0->gensim) (2.12.0)
Requirement already satisfied: fst-pso==1.8.1 in c:\users\ibrah\anaconda3\lib\site
-packages (from pyfume->FuzzyTM>=0.4.0->gensim) (1.8.1)
Requirement already satisfied: miniful in c:\users\ibrah\anaconda3\lib\site-packag
es (from fst-pso==1.8.1->pyfume->FuzzyTM>=0.4.0->gensim) (0.0.6)
```

In [4]:

```python
import zipfile
import re
from transformers import pipeline, AutoTokenizer, AutoModelForSequenceClassificatic
import pandas as pd

# Set up sentiment analysis pipeline
model_name = "distilbert-base-uncased-finetuned-sst-2-english"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)
sentiment_pipeline = pipeline("sentiment-analysis", model=model, tokenizer=tokenize

def extract_customer_transcript(transcript):
    customer_lines = re.findall(r'Member: (.*?)(?=\n|$)', transcript)
    return ' '.join(customer_lines)

def analyze_sentiment(text):
    result = sentiment_pipeline(text[:512])[0]  # Limit to 512 tokens
    if result['label'] == 'POSITIVE' and result['score'] > 0.6:
        return 'positive'
    elif result['label'] == 'NEGATIVE' and result['score'] > 0.6:
        return 'negative'
    else:
        return 'neutral'

def determine_outcome(text):
    positive_indicators = ['thank you', 'great', 'sounds good', 'that\'s all', 'res
    negative_indicators = ['not resolved', 'still have a problem', 'unhappy', 'disa

    text_lower = text.lower()

    positive_count = sum(1 for indicator in positive_indicators if indicator in tex
    negative_count = sum(1 for indicator in negative_indicators if indicator in tex

    if positive_count > negative_count and 'thank you' in text_lower:
        return 'issue resolved'
    elif negative_count > 0 or 'follow up' in text_lower or 'call back' in text_low
        return 'follow-up action needed'
    else:
        return 'unclear'

def extract_call_duration(transcript):
    duration_match = re.search(r'The conversation ends after (\d+) minutes', transc
    return int(duration_match.group(1)) if duration_match else None
```

```
def extract_call_type(transcript):
    if 'pre-authorization' in transcript.lower():
        return 'pre-authorization request'
    elif 'schedule an appointment' in transcript.lower():
        return 'appointment scheduling'
    else:
        return 'general inquiry'

def analyze_transcript(transcript):
    customer_transcript = extract_customer_transcript(transcript)
    sentiment = analyze_sentiment(customer_transcript)
    outcome = determine_outcome(customer_transcript)
    duration = extract_call_duration(transcript)
    call_type = extract_call_type(transcript)
    return sentiment, outcome, duration, call_type

def process_transcripts(zip_file):
    results = []
    with zipfile.ZipFile(zip_file, 'r') as z:
        for filename in z.namelist():
            if filename.endswith('.txt'):
                with z.open(filename) as f:
                    transcript = f.read().decode('utf-8')
                    sentiment, outcome, duration, call_type = analyze_transcript(tr
                    results.append({
                        'filename': filename,
                        'sentiment': sentiment,
                        'outcome': outcome,
                        'duration': duration,
                        'call_type': call_type
                    })
    return pd.DataFrame(results)

def main():
    zip_file = 'transcripts_v3.zip'
    df = process_transcripts(zip_file)

    print(df.head())
    print("\nDataFrame Info:")
    print(df.info())

    print("\nOutcome Distribution:")
    print(df['outcome'].value_counts(normalize=True))

    print("\nSentiment Distribution:")
    print(df['sentiment'].value_counts(normalize=True))

    print("\nCall Type Distribution:")
    print(df['call_type'].value_counts(normalize=True))

    # Save the DataFrame to a CSV file
    df.to_csv('transcript_analysis_results.csv', index=False)
    print("\nResults saved to 'transcript_analysis_results.csv'")

if __name__ == "__main__":
    main()
```

```
                          filename sentiment         outcome  duration  \
0  transcripts_v3/transcript_0.txt  positive  issue resolved       NaN
1  transcripts_v3/transcript_1.txt  negative  issue resolved       NaN
2  transcripts_v3/transcript_2.txt  positive  issue resolved       NaN
3  transcripts_v3/transcript_3.txt  negative  issue resolved       NaN
4  transcripts_v3/transcript_4.txt  positive  issue resolved       NaN

                   call_type
0  pre-authorization request
1            general inquiry
2            general inquiry
3            general inquiry
4     appointment scheduling

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   filename   200 non-null    object
 1   sentiment  200 non-null    object
 2   outcome    200 non-null    object
 3   duration   1 non-null      float64
 4   call_type  200 non-null    object
dtypes: float64(1), object(4)
memory usage: 7.9+ KB
None

Outcome Distribution:
issue resolved             0.92
follow-up action needed    0.06
unclear                    0.02
Name: outcome, dtype: float64

Sentiment Distribution:
positive    0.535
negative    0.435
neutral     0.030
Name: sentiment, dtype: float64

Call Type Distribution:
general inquiry              0.655
appointment scheduling       0.175
pre-authorization request    0.170
Name: call_type, dtype: float64

Results saved to 'transcript_analysis_results.csv'
```

In [5]:

```python
import numpy as np
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

def calculate_class_distribution(df, column):
    distribution = df[column].value_counts(normalize=True)
    return distribution

def calculate_sentiment_outcome_correlation(df):
    contingency_table = pd.crosstab(df['sentiment'], df['outcome'])
    return contingency_table

def plot_confusion_matrix(df, x, y, title):
    cm = confusion_matrix(df[x], df[y], labels=sorted(df[x].unique()))
```

```
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=sorted(df[x].unique()),
                yticklabels=sorted(df[y].unique()))
    plt.title(title)
    plt.xlabel(y)
    plt.ylabel(x)
    plt.savefig(f'{x}_{y}_confusion_matrix.png')
    plt.close()

def calculate_confidence_metrics(df):
    sentiment_scores = df['sentiment'].map({'positive': 1, 'neutral': 0, 'negative'
    outcome_scores = df['outcome'].map({'issue resolved': 1, 'unclear': 0, 'follow-

    sentiment_confidence = np.abs(sentiment_scores)
    outcome_confidence = np.abs(outcome_scores)

    return {
        'avg_sentiment_confidence': sentiment_confidence.mean(),
        'avg_outcome_confidence': outcome_confidence.mean()
    }

def analyze_model_performance(df):
    print("\nModel Performance Metrics:")

    print("\n1. Class Distribution:")
    print("Sentiment Distribution:")
    print(calculate_class_distribution(df, 'sentiment'))
    print("\nOutcome Distribution:")
    print(calculate_class_distribution(df, 'outcome'))
    print("\nCall Type Distribution:")
    print(calculate_class_distribution(df, 'call_type'))

    print("\n2. Sentiment-Outcome Correlation:")
    print(calculate_sentiment_outcome_correlation(df))

    print("\n3. Confusion Matrices:")
    plot_confusion_matrix(df, 'sentiment', 'outcome', 'Sentiment vs Outcome')
    plot_confusion_matrix(df, 'call_type', 'outcome', 'Call Type vs Outcome')
    print("Confusion matrices saved as PNG files.")

    print("\n4. Confidence Metrics:")
    confidence_metrics = calculate_confidence_metrics(df)
    print(f"Average Sentiment Confidence: {confidence_metrics['avg_sentiment_confic
    print(f"Average Outcome Confidence: {confidence_metrics['avg_outcome_confidence

    print("\n5. Potential Biases:")
    print("Outcome by Call Type:")
    print(df.groupby('call_type')['outcome'].value_counts(normalize=True).unstack()

def main():
    zip_file = 'transcripts_v3.zip'
    df = process_transcripts(zip_file)

    print(df.head())
    print("\nDataFrame Info:")
    print(df.info())

    analyze_model_performance(df)

    df.to_csv('transcript_analysis_results.csv', index=False)
    print("\nResults saved to 'transcript_analysis_results.csv'")
```

```
if __name__ == "__main__":
    main()
```

```
                          filename sentiment         outcome  duration  \
0  transcripts_v3/transcript_0.txt  positive  issue resolved       NaN
1  transcripts_v3/transcript_1.txt  negative  issue resolved       NaN
2  transcripts_v3/transcript_2.txt  positive  issue resolved       NaN
3  transcripts_v3/transcript_3.txt  negative  issue resolved       NaN
4  transcripts_v3/transcript_4.txt  positive  issue resolved       NaN

                   call_type
0  pre-authorization request
1            general inquiry
2            general inquiry
3            general inquiry
4     appointment scheduling

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   filename   200 non-null    object
 1   sentiment  200 non-null    object
 2   outcome    200 non-null    object
 3   duration   1 non-null      float64
 4   call_type  200 non-null    object
dtypes: float64(1), object(4)
memory usage: 7.9+ KB
None

Model Performance Metrics:

1. Class Distribution:
Sentiment Distribution:
positive    0.535
negative    0.435
neutral     0.030
Name: sentiment, dtype: float64

Outcome Distribution:
issue resolved             0.92
follow-up action needed    0.06
unclear                    0.02
Name: outcome, dtype: float64

Call Type Distribution:
general inquiry              0.655
appointment scheduling       0.175
pre-authorization request    0.170
Name: call_type, dtype: float64

2. Sentiment-Outcome Correlation:
outcome    follow-up action needed  issue resolved  unclear
sentiment
negative                        11              73        3
neutral                          0               6        0
positive                         1             105        1

3. Confusion Matrices:
Confusion matrices saved as PNG files.

4. Confidence Metrics:
Average Sentiment Confidence: 0.97
Average Outcome Confidence: 0.98
```

```
5. Potential Biases:
Outcome by Call Type:
outcome                    follow-up action needed  issue resolved   unclear
call_type
appointment scheduling                    0.114286        0.885714       NaN
general inquiry                           0.053435        0.916031  0.030534
pre-authorization request                 0.029412        0.970588       NaN

Results saved to 'transcript_analysis_results.csv'
```