IS6713FinalProject

May 2, 2025

IS6713 Final Project Modeling Results

[32]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from textblob import TextBlob
import string
import re

df = pd.read_excel("Annotated_Comments.xlsx")
df.columns = ["Comment", "Technology", "Political"]

df["Label"] = df["Technology"].map({"tech": 1, "NoneTech": 0})

df = df.dropna(subset=["Comment", "Label"])

X = df["Comment"].astype(str)
y = df["Label"]

X_train, X_test, y_train, y_test = train_test_split(
    df['Comment'], df['Label'], test_size=0.2, random_state=42,␣
 ↪stratify=df['Label']
)

tfidf = TfidfVectorizer(stop_words='english', max_features=1000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

def extract_sentiment_features(texts):
    polarity = []
    subjectivity = []
    for text in texts:
        blob = TextBlob(str(text))
        polarity.append(blob.sentiment.polarity)
        subjectivity.append(blob.sentiment.subjectivity)
    return np.array(list(zip(polarity, subjectivity)))

X_train_lex = extract_sentiment_features(X_train)
X_test_lex = extract_sentiment_features(X_test)

def extract_structural_features(texts):
    features = []
    for text in texts:
        text = str(text)
        num_caps = sum(1 for c in text if c.isupper())
        num_exclaims = text.count('!')
        length = len(text)
        features.append([num_caps, num_exclaims, length])
    return np.array(features)

X_train_struct = extract_structural_features(X_train)
X_test_struct = extract_structural_features(X_test)

models = {}
scores = {}

for name, Xtr, Xte in [
    ("TF-IDF", X_train_tfidf, X_test_tfidf),
    ("Lexicon", X_train_lex, X_test_lex),
    ("Structural", X_train_struct, X_test_struct)
]:
    model = LogisticRegression(max_iter=1000)
    model.fit(Xtr, y_train)
    y_pred = model.predict(Xte)
    macro_f1 = f1_score(y_test, y_pred, average='macro')
    micro_f1 = f1_score(y_test, y_pred, average='micro')
    scores[name] = (macro_f1, micro_f1)
    models[name] = (model, Xte, y_pred)

for model_name, (macro, micro) in scores.items():
    print(f"{model_name} - Macro F1: {macro:.4f}, Micro F1: {micro:.4f}")

best_model_name = max(scores.items(), key=lambda x: x[1][0])[0]
print(f"Best model: {best_model_name}")
print("Scores:", scores[best_model_name])

best_model, best_X, best_preds = models[best_model_name]
output_df = pd.DataFrame({
    "comment": X_test,
    "true_label": y_test.values,
    "predicted_label": best_preds
})
```

```
output_df.to_csv("best_model_predictions_tech.csv", index=False)
```

```
TF-IDF - Macro F1: 0.5726, Micro F1: 0.6650
Lexicon - Macro F1: 0.3865, Micro F1: 0.6300
Structural - Macro F1: 0.3990, Micro F1: 0.6300
Best model: TF-IDF
Scores: (0.5725814168607062, 0.665)
```

[33]:
```
df["Label"] = df["Political"].map({"Pol": 1, "NoPol": 0})

df = df.dropna(subset=["Comment", "Label"])

X = df["Comment"].astype(str)
y = df["Label"]

X_train, X_test, y_train, y_test = train_test_split(
    df['Comment'], df['Label'], test_size=0.2, random_state=42,␣
 ↪stratify=df['Label']
)

tfidf = TfidfVectorizer(stop_words='english', max_features=1000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

def extract_sentiment_features(texts):
    polarity = []
    subjectivity = []
    for text in texts:
        blob = TextBlob(str(text))
        polarity.append(blob.sentiment.polarity)
        subjectivity.append(blob.sentiment.subjectivity)
    return np.array(list(zip(polarity, subjectivity)))

X_train_lex = extract_sentiment_features(X_train)
X_test_lex = extract_sentiment_features(X_test)

def extract_structural_features(texts):
    features = []
    for text in texts:
        text = str(text)
        num_caps = sum(1 for c in text if c.isupper())
        num_exclaims = text.count('!')
        length = len(text)
        features.append([num_caps, num_exclaims, length])
    return np.array(features)

X_train_struct = extract_structural_features(X_train)
```

```python
X_test_struct = extract_structural_features(X_test)

models = {}
scores = {}

for name, Xtr, Xte in [
    ("TF-IDF", X_train_tfidf, X_test_tfidf),
    ("Lexicon", X_train_lex, X_test_lex),
    ("Structural", X_train_struct, X_test_struct)
]:
    model = LogisticRegression(max_iter=1000)
    model.fit(Xtr, y_train)
    y_pred = model.predict(Xte)
    macro_f1 = f1_score(y_test, y_pred, average='macro')
    micro_f1 = f1_score(y_test, y_pred, average='micro')
    scores[name] = (macro_f1, micro_f1)
    models[name] = (model, Xte, y_pred)

for model_name, (macro, micro) in scores.items():
    print(f"{model_name} - Macro F1: {macro:.4f}, Micro F1: {micro:.4f}")

best_model_name = max(scores.items(), key=lambda x: x[1][0])[0]
print(f"Best model: {best_model_name}")
print("Scores:", scores[best_model_name])

best_model, best_X, best_preds = models[best_model_name]
output_df = pd.DataFrame({
    "comment": X_test,
    "true_label": y_test.values,
    "predicted_label": best_preds
})
output_df.to_csv("best_model_predictions_pol.csv", index=False)
```

```
TF-IDF - Macro F1: 0.4872, Micro F1: 0.9500
Lexicon - Macro F1: 0.4872, Micro F1: 0.9500
Structural - Macro F1: 0.7217, Micro F1: 0.9650
Best model: Structural
Scores: (0.721725303120652, 0.965)
```