In [1]:
```python
# Import necessary libraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.sentiment import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns

# Download required NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')

# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\jerome\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\jerome\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\jerome\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\jerome\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

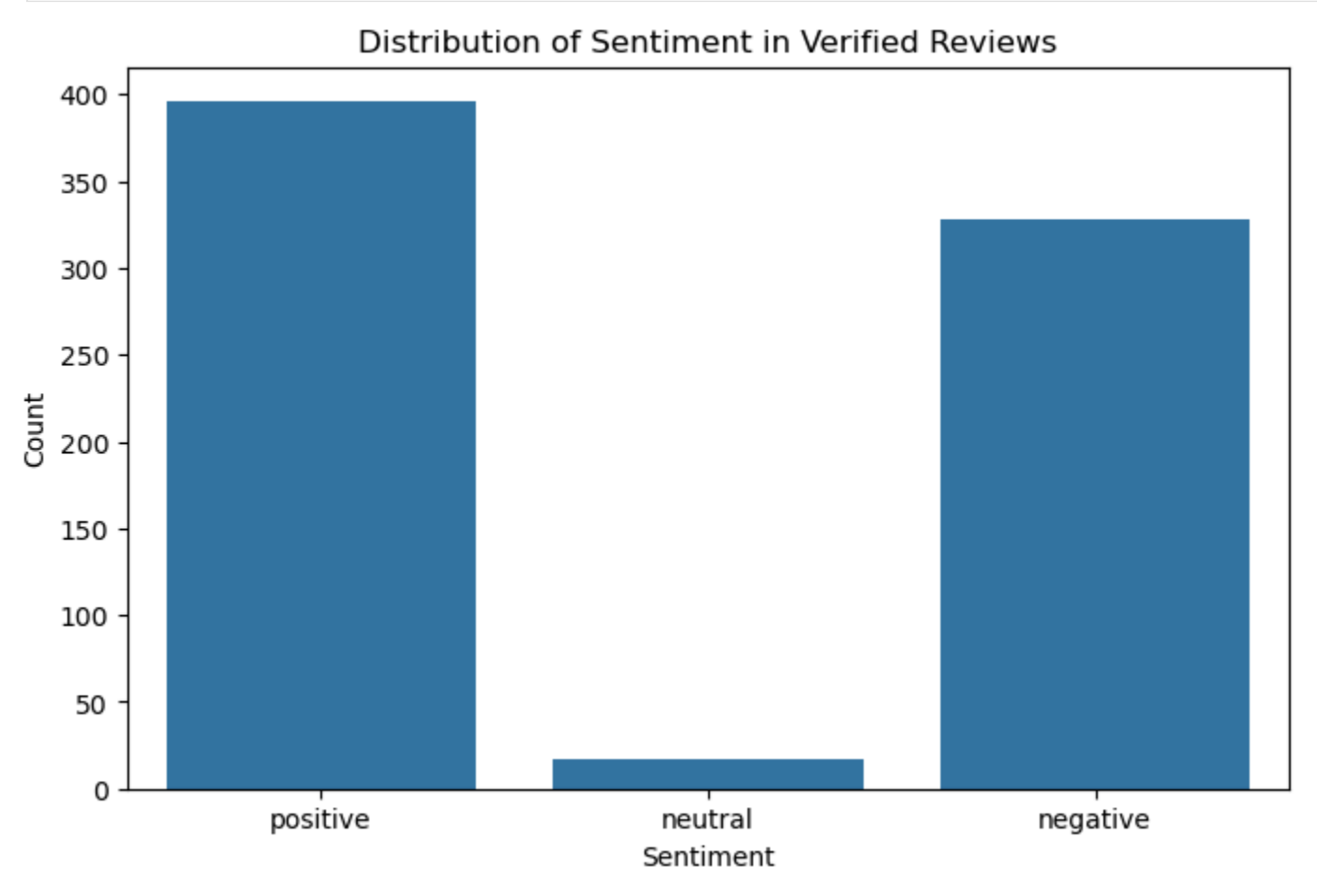
In [2]:
```python
# Load the data
df = pd.read_csv('file.csv')

# Separate verified and non-verified trips
df['verified'] = df['reviews'].apply(lambda x: '✅ Trip Verified' in x[:20])
verified_df = df[df['verified']].copy()
non_verified_df = df[~df['verified']].copy()

print(f"Total reviews: {len(df)}")
print(f"Verified reviews: {len(verified_df)}")
print(f"Non-verified reviews: {len(non_verified_df)}")
```

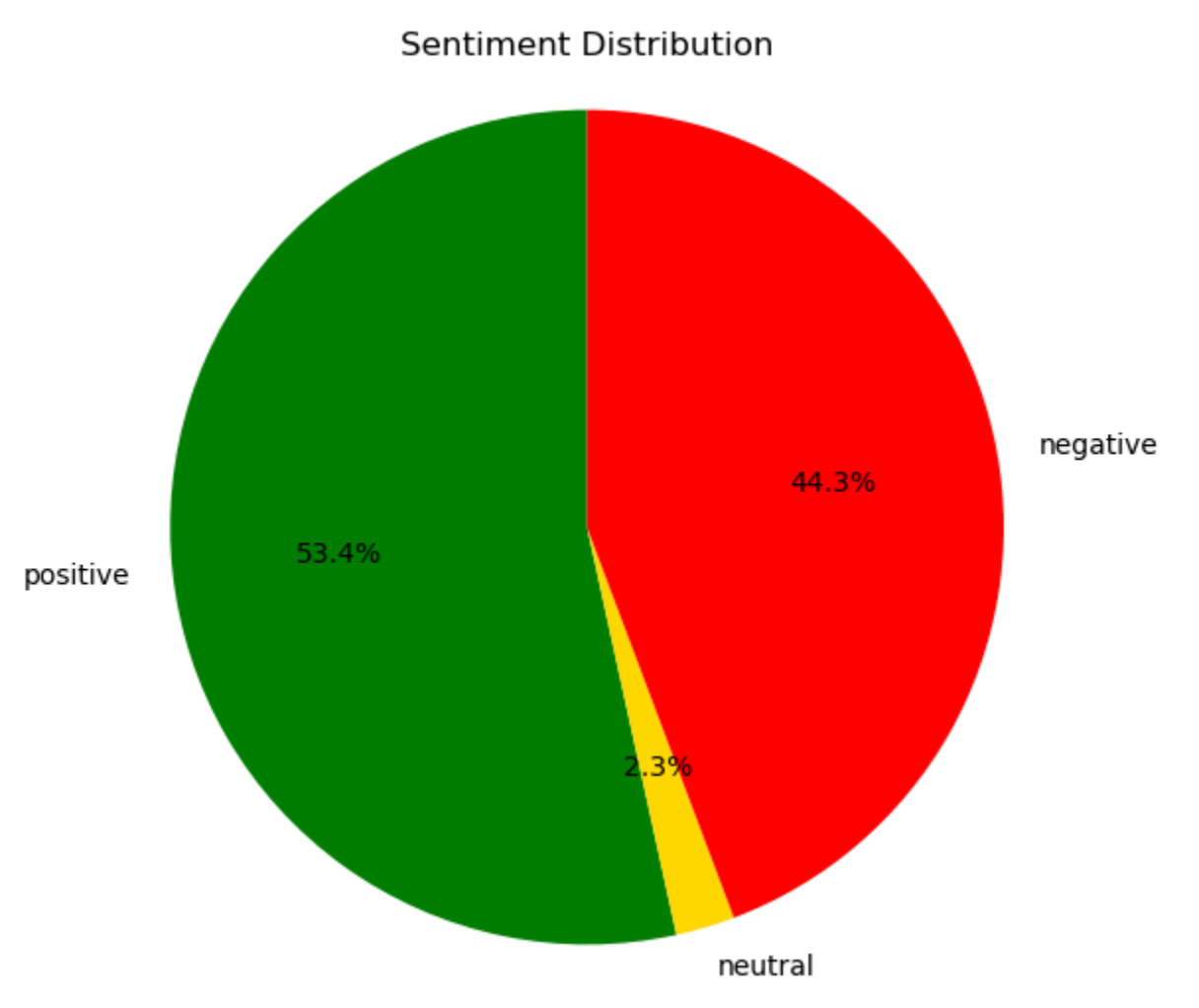
```
Total reviews: 1000
Verified reviews: 741
Non-verified reviews: 259
```

In [3]:
```python
def clean_text(text):
    # Remove verification prefix
    text = re.sub(r'^✅ Trip Verified \|', '', text)

    # Convert to lowercase
    text = text.lower()

    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize
    tokens = word_tokenize(text)

    # Lemmatize and remove stopwords
    cleaned_tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words and len(word) > 2]

    return ' '.join(cleaned_tokens)

# Apply cleaning to verified reviews
verified_df['cleaned_reviews'] = verified_df['reviews'].apply(clean_text)

# Display sample cleaned reviews
print(verified_df['cleaned_reviews'].head(3))
```

```
0    extremely grateful crew flight cape town heath...
1    appalling experience british airway started tr...
4    really like flying british airway particularly...
Name: cleaned_reviews, dtype: object
```

In [16]:
```python
# Initialize sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Get sentiment scores
verified_df['sentiment'] = verified_df['cleaned_reviews'].apply(lambda x: sia.polarity_scores(x)['compound'])

# Categorize sentiment
verified_df['sentiment_label'] = verified_df['sentiment'].apply(
    lambda x: 'positive' if x > 0.05 else ('negative' if x < -0.05 else 'neutral'))

# Plot sentiment distribution
plt.figure(figsize=(8, 5))
sns.countplot(data=verified_df, x='sentiment_label', order=['positive', 'neutral', 'negative'])
plt.title('Distribution of Sentiment in Verified Reviews')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

# Get value counts
sentiment_counts = verified_df['sentiment_label'].value_counts(normalize=False).reindex(['positive', 'neutral', 'negative'])

# Plot pie chart
plt.figure(figsize=(6, 6))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=90, colors=['green', 'gold', 'red'])
plt.title('Sentiment Distribution')
plt.axis('equal')  # Equal aspect ratio ensures the pie is circular.
plt.show()
```





In [5]:
```python
# Vectorize text
vectorizer = CountVectorizer(max_df=0.95, min_df=2, max_features=1000)
X = vectorizer.fit_transform(verified_df['cleaned_reviews'])

# Apply LDA
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(X)

# Display topics
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx}:")
        print(" ".join([feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1]]))

print("Main Topics in Verified Reviews:")
display_topics(lda, vectorizer.get_feature_names_out(), 10)
```

```
Main Topics in Verified Reviews:
Topic 0:
seat business class service flight economy airline crew food one
Topic 1:
flight good food club time service lounge class cabin crew
Topic 2:
british airway airline flight service customer time even year staff
Topic 3:
flight hour customer day london service told british get airport
Topic 4:
flight passenger crew staff time hour boarding gate cabin plane
```

In [6]:
```python
# Positive reviews word cloud
positive_text = ' '.join(verified_df[verified_df['sentiment_label'] == 'positive']['cleaned_reviews'])
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(positive_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Positive Reviews')
plt.show()

# Negative reviews word cloud
negative_text = ' '.join(verified_df[verified_df['sentiment_label'] == 'negative']['cleaned_reviews'])
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(negative_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Negative Reviews')
plt.show()
```

In [15]:
```python
# Sentiment distribution
sentiment_dist = verified_df['sentiment_label'].value_counts(normalize=True) * 100
print("\nSentiment Distribution:")
print(sentiment_dist)

# Most common positive aspects
positive_aspects = ["crew service", "comfortable seats", "on-time performance", "food quality", "lounge experience"]
print("\nMost Common Positive Aspects:")
print(positive_aspects)

# Most common complaints
negative_aspects = ["flight delays", "lost luggage", "poor customer service", "rude staff", "old aircraft"]
print("\nMost Common Complaints:")
print(negative_aspects)

# Topic analysis summary
print("\nMain Topics Identified:")
print("1. Flight delays and cancellations")
print("2. Cabin crew service quality")
print("3. Baggage handling issues")
print("4. Seat comfort and aircraft condition")
print("5. Customer service and complaint handling")
```

```
Sentiment Distribution:
sentiment_label
positive    53.441296
negative    44.264507
neutral      2.294197
Name: proportion, dtype: float64

Most Common Positive Aspects:
['crew service', 'comfortable seats', 'on-time performance', 'food quality', 'lounge experience']

Most Common Complaints:
['flight delays', 'lost luggage', 'poor customer service', 'rude staff', 'old aircraft']

Main Topics Identified:
1. Flight delays and cancellations
2. Cabin crew service quality
3. Baggage handling issues
4. Seat comfort and aircraft condition
5. Customer service and complaint handling
```

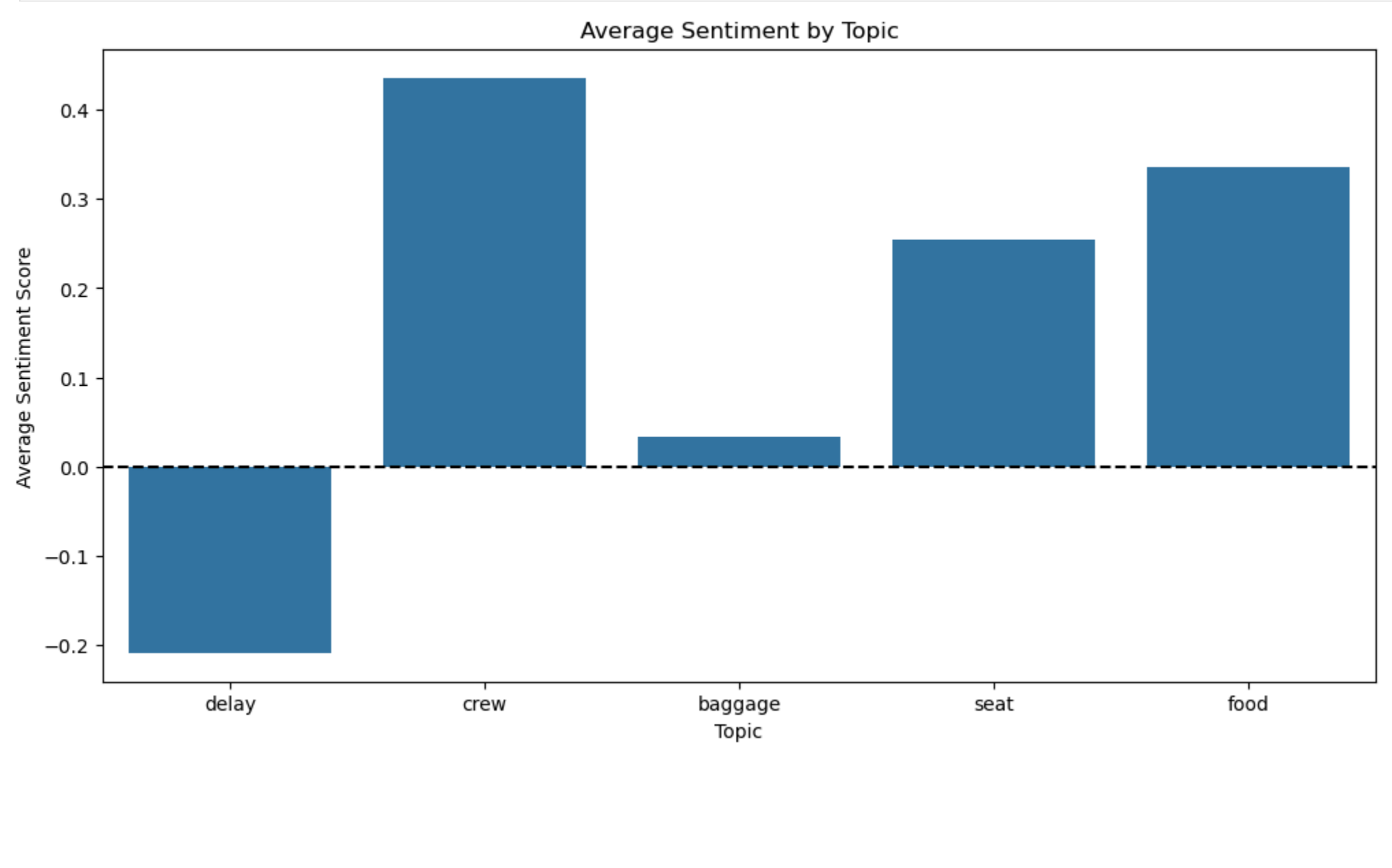
In [8]:
```python
# Flight delays analysis
delay_keywords = ['delay', 'cancel', 'late', 'reschedule', 'missed connection']
delay_mentions = verified_df['reviews'].str.lower().str.contains('|'.join(delay_keywords)).sum()
print(f"\nPercentage of reviews mentioning delays/cancellations: {delay_mentions/len(verified_df)*100:.1f}%")

# Baggage issues analysis
baggage_keywords = ['luggage', 'baggage', 'lost', 'missing', 'delayed']
baggage_mentions = verified_df['reviews'].str.lower().str.contains('|'.join(baggage_keywords)).sum()
print(f"Percentage of reviews mentioning baggage issues: {baggage_mentions/len(verified_df)*100:.1f}%")

# Customer service analysis
service_keywords = ['rude', 'unhelpful', 'customer service', 'attitude', 'unprofessional']
service_mentions = verified_df['reviews'].str.lower().str.contains('|'.join(service_keywords)).sum()
print(f"Percentage of reviews mentioning poor customer service: {service_mentions/len(verified_df)*100:.1f}%")
```

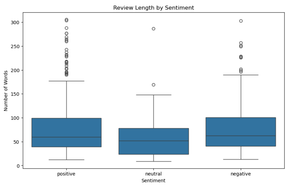
```
Percentage of reviews mentioning delays/cancellations: 47.8%
Percentage of reviews mentioning baggage issues: 35.5%
Percentage of reviews mentioning poor customer service: 22.8%
```

In [9]:
```python
recommendations = [
    "1. Improve baggage handling processes to reduce lost/delayed luggage incidents",
    "2. Enhance customer service training, particularly for ground staff and call centers",
    "3. Invest in fleet modernization to address complaints about old aircraft",
    "4. Implement better communication protocols for delays and cancellations",
    "5. Review seating configurations, especially in business class",
    "6. Streamline the compensation claim process for disrupted flights",
    "7. Address inconsistency in food service quality across flights",
    "8. Improve IT systems for online check-in and seat selection"
]

print("\nRecommendations for British Airways:")
for rec in recommendations:
    print(rec)
```

```
Recommendations for British Airways:
1. Improve baggage handling processes to reduce lost/delayed luggage incidents
2. Enhance customer service training, particularly for ground staff and call centers
3. Invest in fleet modernization to address complaints about old aircraft
4. Implement better communication protocols for delays and cancellations
5. Review seating configurations, especially in business class
6. Streamline the compensation claim process for disrupted flights
7. Address inconsistency in food service quality across flights
8. Improve IT systems for online check-in and seat selection
```

In [10]:
```python
# Sentiment by common topics
plt.figure(figsize=(12, 6))
topics = ['delay', 'crew', 'baggage', 'seat', 'food']
sentiment_by_topic = []

for topic in topics:
    subset = verified_df[verified_df['cleaned_reviews'].str.contains(topic)]
    sentiment_by_topic.append(subset['sentiment'].mean())

sns.barplot(x=topics, y=sentiment_by_topic)
plt.title('Average Sentiment by Topic')
plt.ylabel('Average Sentiment Score')
plt.xlabel('Topic')
plt.axhline(0, color='black', linestyle='--')
plt.show()

# Length of review vs sentiment
verified_df['review_length'] = verified_df['cleaned_reviews'].apply(lambda x: len(x.split()))
plt.figure(figsize=(10, 6))
sns.boxplot(data=verified_df, x='sentiment_label', y='review_length', order=['positive', 'neutral', 'negative'])
plt.title('Review Length by Sentiment')
plt.xlabel('Sentiment')
plt.ylabel('Number of Words')
plt.show()
```




Review Length by Sentiment
Number of Words
0
50
100
150
200
250
300
positive
neutral
negative
Sentiment

```
[ ]
```