Analyse e-Réputationnelle de la Franchise Tim Hortons

1 Introduction

Ce rapport décrit la méthodologie utilisée pour effectuer une analyse e-réputationnelle de la franchise Tim Hortons à partir des avis en ligne. L'objectif est de comprendre les sentiments exprimés par les clients, identifier les points positifs et négatifs, et fournir des recommandations basées sur ces analyses.

2 Étapes de la Méthodologie

2.1 Chargement et Préparation des Données

Bibliothèques utilisées :

- `streamlit`
- `pandas`
- `numpy`

Les avis des clients ont été chargés à partir d'un fichier CSV en utilisant `pandas`. Les notes (ratings) ont été nettoyées et converties en format numérique.

```
data_path = 'review.csv'
Data = pd.read_csv(data_path, encoding='utf-8')
Data['Rating'] = pd.to_numeric(Data['Rating'].str.replace
('etoile', '').str.replace('étoiles', '').str.strip(), errors='coerce')
Data['Rating'] = Data['Rating'].astype('Int64', errors='ignore')
```

2.2 Analyse des Sentiments

Bibliothèques utilisées :

- `transformers`
- `vaderSentiment`

L'analyse des sentiments a été réalisée à l'aide de deux outils :

1. **Pipeline d'analyse des sentiments** de la bibliothèque `transformers` pour classifier les avis en "Positive", "Negative" ou "Neutral".
2. **VADER Sentiment** pour une analyse fine des sentiments en utilisant des règles prédéfinies.

```
sentiment_pipeline = pipeline("sentiment-analysis")

def get_sentiment_advanced(review):
    if isinstance(review, str):
        result = sentiment_pipeline(review[:512])
        sentiment = result[0]['label']
        return 'Positive' if sentiment == 'POSITIVE' else 'Negative'
        if sentiment == 'NEGATIVE' else 'Neutral'
    else:
        return 'Neutral'

Data['Review_english'] = Data['Review_english'].astype(str)
Data['Sentiment'] = Data['Review_english'].apply(get_sentiment_advanced)
```

2.3 Visualisation des Sentiments

Bibliothèques utilisées :

- `matplotlib`

Les résultats de l'analyse des sentiments ont été visualisés sous forme de diagrammes à barres pour montrer la distribution des sentiments et des notes.

```
fig, ax = plt.subplots(figsize=(10, 6))
sentiment_counts.plot(kind='bar', color='blue', ax=ax)
ax.set_xlabel('Sentiment')
ax.set_ylabel('Percentage')
ax.set_title('Distribution of Sentiments')
ax.grid(axis='y')
st.pyplot(fig)
```

2.4 Analyse des Mots Fréquents

Bibliothèques utilisées :

- `nltk`
- `wordcloud`
- `collections.Counter`

Les avis ont été nettoyés pour enlever la ponctuation et les stopwords, puis les mots fréquents ont été identifiés à l'aide de `Counter`. Un nuage de mots a été généré pour visualiser les mots les plus utilisés dans les avis.

```
all_reviews = ' '.join([str(review) for review in Data['Review_english']])
words = all_reviews.split()
stop_words = set(stopwords.words('english'))
filtered_words = [word for word in words if word.lower() not in stop_words]
word_counts = Counter(filtered_words)

wordcloud = WordCloud(width=800, height=400, background_color='white').generate
(' '.join(filtered_words))
fig, ax = plt.subplots(figsize=(10, 5))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis('off')
st.pyplot(fig)
```

2.5 Extraction des Noms et Correction Orthographique

Bibliothèques utilisées :

- `textblob`

Les noms (nouns) ont été extraits des avis pour une analyse plus détaillée. La correction orthographique a été appliquée pour améliorer la qualité des textes analysés.

```
def extract_nouns(review):
    if isinstance(review, str):
        blob = TextBlob(review)
        pos_tags = blob.tags
        nouns = [word for word, pos in pos_tags if pos in ['NN', 'NNS']]
        return ' '.join(nouns)
    else:
        return ''

def correct_spelling(text):
    if isinstance(text, str):
        blob = TextBlob(text)
        return str(blob.correct())
    else:
        return ''

Data['Review_english'] = Data['Review_english'].apply(correct_spelling)
Data['Nouns'] = Data['Review_english'].apply(extract_nouns)
```

2.6 Analyse des Sentiments par Catégorie

Les avis ont été filtrés par mots-clés spécifiques à certaines catégories (ex. personnel, nourriture, propreté) et l'analyse des sentiments a été appliquée à chaque catégorie.

```
keywords = {
    'staff': ['staff', 'cashier', 'employees', 'employee', 'workers', 'service',
    'attitude'],
    'cuisine': ['food', 'portions', 'cappuccino', 'chocolate', 'cuisine', 'repas', 'plat',
    'coffee', 'drinks', 'breakfast', 'lunch', 'meal', 'milk', 'dinner', 'tea', 'cinnamon',
    'sandwich', 'express', 'vanilla', 'menu', 'taste'],
    'cleanliness': ['clean', 'bathroom', 'machine', 'cockroaches', 'propreté', 'propre',
    'sale', 'hygienic', 'dirty', 'cleaning', 'toilet'],
}

def analyze_sentiment_by_category(data, keywords):
    for category, words in keywords.items():
        data[category + '_filtered'] = data['Review_english'].apply
        (lambda review:filter_reviews_by_keywords(review, words))
        data[category + '_sentiment'] = data[category + '_filtered'].apply
        (get_sentiment_vader)
    return data

Data = analyze_sentiment_by_category(Data, keywords)
```

2.7 Conversion des Dates et Analyse Temporelle

Les avis ont été convertis en jours pour une analyse temporelle de la distribution des sentiments.

```
import re
from datetime import datetime, timedelta

def convert_to_days(date_str):
    date_str = date_str.replace('un', '1').replace('une', '1')

    today = datetime.now()

    match = re.search(r'(\d+)\s*(jour|jours|mois|an|année|années)', date_str)
    if match:
        num = int(match.group(1))
        unit = match.group(2)

        if 'jour' in unit:
            return num
        elif 'mois' in unit:
            past_date = today - pd.DateOffset(months=num)
            return (today - past_date).days
        elif 'an' in unit or 'année' in unit or 'années' in unit:
            return num * 365
    return None
```

```
Data['Date_in_days'] = Data['Date'].apply(convert_to_days)

reviews_by_date_sorted = Data.groupby('Date_in_days').apply
(lambda x:x['Sentiment'].value_counts()).unstack().fillna(0).sort_index()

fig, ax = plt.subplots(figsize=(12, 6))
reviews_by_date_sorted.plot(kind='bar', stacked=True, ax=ax)
ax.set_title('Sentiment Analysis of Reviews Over Time')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Reviews')
ax.set_xticks(reviews_by_date_sorted.index)
ax.set_xticklabels(reviews_by_date_sorted.index, rotation=45)
ax.legend(title='Sentiment')
ax.grid(True)
st.pyplot(fig)
```

2.8 Comparaison Entre Franchises

Les avis de deux franchises ont été comparés pour analyser les différences de sentiments.

```
data_franchise1 =
pd.read_csv('review.csv')
data_franchise2 = pd.read_csv('review_peel.csv')

def get_sentiment(review):
    if isinstance(review, str):
        result = sentiment_analysis(review[:512])
        sentiment = result[0]['label']
        return 'Positive' if sentiment == 'POSITIVE' else 'Negative'
        if sentiment == 'NEGATIVE' else 'Neutral'
    else:
        return 'Neutral'

data_franchise1['Sentiment'] = data_franchise1['Review_english'].apply(get_sentiment)
data_franchise2['Sentiment'] = data_franchise2['Review_english'].apply(get_sentiment)

sentiments_franchise1 = data_franchise1['Sentiment'].value_counts(normalize=True)
sentiments_franchise2 = data_franchise2['Sentiment'].value_counts(normalize=True)

sentiments = pd.DataFrame({
    'Franchise 1': sentiments_franchise1,
    'Franchise 2': sentiments_franchise2
}).fillna(0)
```

```
fig, ax = plt.subplots(figsize=(10, 6))
sentiments.plot(kind='bar', ax=ax)
ax.set_title('Comparaison des Sentiments des Avis entre deux Franchises')
ax.set_ylabel('Proportion des Avis')
st.pyplot(fig)

data_franchise1['Date_in_days'] = data_franchise1['Date'].apply(convert_to_days)
data_franchise2['Date_in_days'] = data_franchise2['Date'].apply(convert_to_days)

data_franchise1['Category'] = data_franchise1['Review'].apply(get_sentiment_vader)
data_franchise2['Category'] = data_franchise2['Review'].apply(get_sentiment_vader)

data_franchise1.dropna(subset=['Date_in_days'], inplace=True)
data_franchise2.dropna(subset=['Date_in_days'], inplace=True)

reviews_by_category_date1 = data_franchise1.groupby(['Date_in_days', 'Category']).
apply(lambda x: x['Sentiment'].value_counts()).unstack().fillna(0)
reviews_by_category_date2 = data_franchise2.groupby(['Date_in_days', 'Category']).
apply(lambda x: x['Sentiment'].value_counts()).unstack().fillna(0)

reviews_by_category_date1.to_csv('reviews_by_category_date1.csv')
reviews_by_category_date2.to_csv('reviews_by_category_date2.csv')

categories = keywords.keys()
for category in categories:
    if category in reviews_by_category_date1.index.get_level_values('Category'):
        fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(15, 14), sharex=True)

        category_data1 = reviews_by_category_date1.xs(category, level='Category',
        drop_level=False)
        category_data1.plot(kind='bar', stacked=True, ax=axes[0],

        title=f"Sentiments over time for {category} (Franchise 1)")
        axes[0].set_ylabel('Number of Reviews')
        axes[0].set_xlabel('Date in Days')

        category_data2 = reviews_by_category_date2.xs(category,
        level='Category', drop_level=False)
        category_data2.plot(kind='bar', stacked=True, ax=axes[1],
        title=f"Sentiments over time for {category} (Franchise 2)")
        axes[1].set_ylabel('Number of Reviews')
        axes[1].set_xlabel('Date in Days')

        st.pyplot(fig)
```

2.9 Analyse des Avis Négatifs

Les avis négatifs ont été extraits pour identifier les thèmes récurrents et proposer des actions concrètes.

```
negative_reviews = data_franchise1[data_franchise1['Sentiment'] == 'Negative']
['Review_english']

vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(negative_reviews)
word_counts = Counter(vectorizer.get_feature_names_out())

common_words = word_counts.most_common(10)
st.write("Mots les plus fréquents dans les avis négatifs:")
st.write(common_words)

recommendations = {
    "service": "Former le personnel pour améliorer l'interaction avec les clients.",
    "qualité": "Assurer un contrôle de qualité plus rigoureux des produits.",
    "attente": "Optimiser les processus pour réduire le temps d'attente des clients.",
}

priority = {
    "service": 1,
    "qualité": 2,
    "attente": 3
}

df_recommendations = pd.DataFrame(list(recommendations.items()),
columns=['Problème', 'Recommandation'])
df_recommendations['Priorité'] = df_recommendations['Problème'].map(priority)

df_recommendations.sort_values(by='Priorité', inplace=True)

st.write(df_recommendations)

df_recommendations.to_csv('recommendations.csv', index=False, encoding='utf-8')
st.write("Les recommandations ont été enregistrées dans le fichier recommendations.csv")
```

3 Conclusion

La méthodologie décrite ci-dessus permet une analyse détaillée des avis en ligne pour évaluer la réputation d'une franchise. En utilisant des outils avancés de traitement du langage naturel et de visualisation, nous avons pu extraire des informations précieuses sur les sentiments des clients, identifier les aspects les plus discutés, et proposer des recommandations pour améliorer la satisfaction

des clients. Les résultats de cette analyse peuvent aider à cibler les améliorations et à renforcer la réputation de la franchise Tim Hortons.