Comprehensive Course on Sentiment Analysis

Table of Contents

This course builds on **feature extraction techniques (BoW, TF-IDF)** and now dives into **Sentiment Analysis**, explaining: ✅**What** is sentiment analysis?
✅**Why** do we use it?
✅**Where** is it applied?
✅**When** is it useful?
✅**How** do we perform sentiment analysis?

We will also include **a case study**, **practical implementation in Python**, and **challenges in real-world sentiment analysis**.

What is Sentiment Analysis?

Sentiment analysis is a **Natural Language Processing (NLP)** technique that determines the **emotional tone** behind a piece of text. It **classifies opinions as positive, negative, or neutral**.

Example:

- "I love this product! It's amazing!" → **Positive** ?
- "The service was terrible. I will never come back!" → **Negative** ?
- "The movie was okay, nothing special." → **Neutral** ?

Why Do We Use Sentiment Analysis?

1□Understanding Customer Feedback

- **Businesses use it** to analyze **product reviews** and customer feedback.
- Helps in **brand monitoring** and customer satisfaction analysis.

2□Social Media Monitoring

- Companies track **public opinion** on Twitter, Facebook, Reddit, etc.
- Helps identify **trending topics and potential crises**.

3□Market & Competitor Analysis

- Analyzing reviews of **competitor products** helps in **strategy planning**.
- **Stock markets** use it to track **public sentiment on companies**.

4□Employee & Workforce Analysis

- Sentiment analysis can be used to analyze **employee satisfaction surveys**.
- Helps HR teams **improve workplace culture**.

5□Government & Politics

- Used for **analyzing public opinions** on policies, elections, and political speeches.
- Helps in predicting **voter sentiment**.

Where is Sentiment Analysis Applied?

✅**E-commerce & Retail** → Amazon, eBay, Shopify analyze product reviews.
✅**Social Media & News** → Twitter, Facebook monitor public sentiment.
✅**Finance & Stock Market** → Sentiment-based trading strategies.
✅**Healthcare** → Patient feedback analysis.
✅**Entertainment** → Movie, game, music reviews.

When Should You Use Sentiment Analysis?

✔ When you need to **analyze customer feedback at scale**.
✔ When you want to **track brand reputation** in real-time.
✔ When you need **insights from social media and product reviews**.
✔ When you want to **predict trends based on public sentiment**.

How to Perform Sentiment Analysis?

We follow **7 key steps**:

1□Goal Setting

- Define the purpose: Are you analyzing **product reviews, social media, or survey feedback**?

2□Text Data Cleaning

- Remove unnecessary elements:
  ✅Remove **punctuation**
  ✅Remove **stopwords**
  ✅Convert text to **lowercase**
  ✅Remove **emojis, hashtags, mentions**

3□Text Preprocessing

- **Tokenization** → Split text into words.
- **Lemmatization** → Convert words to their base form (e.g., "running" → "run").

4□Sentiment Scoring

- Assign **sentiment polarity** (positive/negative/neutral).
- Use **lexicon-based methods (TextBlob, VADER)** or **Machine Learning models**.

5□Feature Extraction

- Convert text into numerical format using:

✅**Bag of Words (BoW)**
✅**TF-IDF**
✅**Word Embeddings (Word2Vec, BERT)**

6□Model Training

- Train **Machine Learning models** (Naïve Bayes, SVM, LSTMs, Transformers).

7□Model Evaluation

- Evaluate performance using **accuracy, precision, recall, F1-score**.

Sentiment Analysis Using TextBlob (Lexicon-Based Approach)

TextBlob is a simple NLP library built on **NLTK** that can perform sentiment analysis.

Example in Python

```
from textblob import TextBlob

# Sample text
text = "The product is absolutely amazing! I love it."

# Create TextBlob object
blob = TextBlob(text)

# Get sentiment polarity & subjectivity
polarity = blob.sentiment.polarity  # -1 (negative) to +1 (positive)
subjectivity = blob.sentiment.subjectivity  # 0 (objective) to 1 (subjective)

print(f"Sentiment Polarity: {polarity}")
print(f"Subjectivity Score: {subjectivity}")
```

Output

```
Sentiment Polarity: 0.625  (Positive)
Subjectivity Score: 0.75   (Highly opinionated)
```

Case Study: Twitter Sentiment Analysis

Scenario

A company wants to **analyze customer sentiment** on Twitter regarding their latest product launch.

They will: ✔ **Collect tweets** using Twitter API
✔ **Clean and preprocess text**
✔ **Use TextBlob for sentiment analysis**
✔ **Visualize sentiment trends**

Challenges

- **Noise in tweets** (emojis, hashtags, URLs).
- **Sarcasm detection** is difficult.
- **Short texts** make understanding sentiment harder.

Python Implementation

```
import pandas as pd
from textblob import TextBlob

# Sample dataset (tweets)
data = {'tweet': [
    "I love the new iPhone! The camera is fantastic! □ □ □",
    "The customer service was terrible, never buying from them again.",
    "Not sure how I feel about this update. It's okay, I guess."
]}

# Convert to DataFrame
df = pd.DataFrame(data)

# Function to get sentiment polarity
def get_sentiment(text):
    return TextBlob(text).sentiment.polarity

# Apply sentiment analysis
df['sentiment'] = df['tweet'].apply(get_sentiment)

# Categorize sentiment
df['sentiment_label'] = df['sentiment'].apply(lambda x: 'Positive' if x > 0 else ('Negative' if x < 0 else 'Neutral'))

# Display results
print(df)
```

Sample Output

Tweet	Sentiment Score	Sentiment Label
"I love the new iPhone!"	**0.9**	**Positive**
"The customer service was terrible."	**-0.7**	**Negative**
"Not sure how I feel about this update."	**0.1**	**Neutral**

Creating a Sentiment Classification Model

If lexicon-based methods (like TextBlob) are **not enough**, we train a **Machine Learning model**.

Steps to Build an ML Sentiment Model

1 **Collect Data** → Use **IMDB reviews, Twitter data, customer feedback**.
2 **Preprocess Text** → Tokenization, Stopword removal, Lemmatization.
3 **Feature Extraction** → Use **TF-IDF or Word Embeddings**.
4 **Train a Model** → Naïve Bayes, Logistic Regression, or LSTM.
5 **Evaluate Performance** → Accuracy, Precision, Recall, F1-score.

Python Implementation (Naïve Bayes Classifier)

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

# Sample dataset
X = ["I love this product!", "This is the worst thing ever!", "It's okay, not great."]
y = ["Positive", "Negative", "Neutral"]

# Convert text into numerical features
vectorizer = TfidfVectorizer()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naïve Bayes Model
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(X_train, y_train)

# Predict on test data
print(model.predict(["This product is fantastic!"]))  # Output: ['Positive']
print(model.predict(["I hate this service."]))  # Output: ['Negative']
```

Summary

Sentiment Analysis helps classify text as positive, negative, or neutral.
TextBlob is a simple way to analyze sentiment using lexicons.
ML models (Naïve Bayes, SVM, LSTMs) provide **better accuracy**.
Real-world challenges include sarcasm, short texts, and context understanding.

Understanding Sentiment Analysis with Subjectivity

When analyzing text sentiment, we get **two important scores**: 1 **Polarity (Sentiment Score)** → Ranges from **-1 (Negative) to +1 (Positive)**
2 **Subjectivity Score** → Ranges from **0 (Objective) to 1 (Subjective)**

Polarity tells us how positive, neutral, or negative a sentence is.
Subjectivity tells us whether the statement is based on **facts (objective)** or **opinions (subjective)**.

How Polarity & Subjectivity Work Together

Sentence	Polarity (Sentiment)	Subjectivity
"I love this phone, it's amazing!"	**0.9 (Positive)**	**0.8 (Opinion-based)**

Sentence	Polarity (Sentiment)	Subjectivity
"The phone is functional and has a 12MP camera."	**0.0 (Neutral)**	**0.2 (Fact-based)**
"The worst product ever! Waste of money."	**-0.9 (Negative)**	**0.7 (Highly Opinionated)**

➡ **High Subjectivity** means the review is **opinion-based**, useful for understanding emotions.
➡ **Low Subjectivity** means the statement is **factual**, useful for extracting key product features.

Using Sentiment & Subjectivity for Feature Selection

Why is Sentiment a Good Feature Selector?

- **Sentiment acts as a powerful filter** to **reduce noise** in text data.
- We can **remove neutral or low-subjectivity text** that doesn't provide strong signals.
- **Combining sentiment & subjectivity helps in feature extraction**, especially in ML models.

Example: Selecting Features for Product Reviews

Dataset

Review	Polarity	Subjectivity	Feature Selection?
"Amazing phone with a great battery life!"	**0.85 (Positive)**	**0.9 (Highly subjective)**	✅ Keep
"The phone has 128GB storage and 5G support."	**0.0 (Neutral)**	**0.1 (Fact-based)**	❌ Remove
"Horrible customer service! Never buying again."	**-0.85 (Negative)**	**0.8 (Highly subjective)**	✅ Keep

✔ **Remove neutral & fact-based reviews** (low-subjectivity).
✔ **Keep highly subjective reviews** (stronger emotional signals).

Python Implementation: Filtering Features Based on Sentiment & Subjectivity

Example: Keeping Only Strongly Opinionated Text

```
from textblob import TextBlob
import pandas as pd

# Sample dataset
reviews = [
```

```
    "The phone is great, I love the battery life!",  # Positive & Opinionated
    "This phone has a 5000mAh battery and 128GB storage.",  # Neutral & Factual
    "Worst experience ever, the screen cracked in a week!",  # Negative & Opinionated
    "The phone is available in 3 colors: black, white, and blue."  # Neutral & Factual
]

# Create DataFrame
df = pd.DataFrame(reviews, columns=["Review"])

# Function to get polarity & subjectivity
def analyze_sentiment(text):
    blob = TextBlob(text)
    return blob.sentiment.polarity, blob.sentiment.subjectivity

# Apply sentiment analysis
df[['Polarity', 'Subjectivity']] = df['Review'].apply(lambda x: pd.Series(analyze_sentiment(x)))

# Filter: Keep only strong opinions (subjectivity > 0.5) and non-neutral sentiment
df_filtered = df[(df['Subjectivity'] > 0.5) & (df['Polarity'].abs() > 0.2)]

# Show results
import ace_tools as tools
tools.display_dataframe_to_user(name="Filtered Sentiment Data", dataframe=df_filtered)
```

Summary

Sentiment + Subjectivity can act as a filter to remove uninformative data.
Only highly subjective and strong sentiment text provides useful insights.
Feature selection using sentiment improves ML models by focusing on **meaningful text**.