# 1 Sentimental Analysis – Light code demo project by Rohan Pius

```
# This is a simple Python script that demonstrates how to perform a sentiment analysis task.
# It uses the popular spaCy library, which is known for its efficiency and ease of use.
# Sentiment analysis is the process of determining the emotional tone behind a piece of text.

# First, we need to install the necessary libraries and models.
# You can do this by running the following commands in your terminal:
# pip install spacy
# python -m spacy download en_core_web_sm
# The 'en_core_web_sm' is a small English language model for spaCy.

import spacy

# Load the small English language model for spaCy.
# This model includes pre-trained components for part-of-speech tagging,
# named-entity recognition, and more.
try:
    nlp = spacy.load("en_core_web_sm")
except OSError:
    print("The 'en_core_web_sm' model is not downloaded.")
    print("Please run 'python -m spacy download en_core_web_sm' in your terminal.")
    exit()

# We'll create a list of movie reviews to analyze.
# These reviews have a clear sentiment (positive, negative, or neutral).
movie_reviews = [
    "I absolutely loved this film! The acting was superb and the plot was engaging.",
    "This movie was a total disappointment. It was boring and the ending made no sense.",
    "The film had its moments, but overall, it was just average.",
    "I was on the edge of my seat! What a thrilling experience!",
    "The plot was confusing and the characters were not believable. I would not recommend it.",
    "The special effects were fantastic, but the story was weak.",
    "It's a cinematic masterpiece that will be remembered for years to come.",
    "The acting was wooden and the dialogue was cringeworthy.",
    "I have never been so bored in my life. A complete waste of time.",
    "An uplifting and heartwarming story that everyone should see.",
]

# We'll manually classify each review for training and evaluation purposes.
# In a real-world scenario, this data would come from a labeled dataset.
sentiments = [
    "Positive",
    "Negative",
    "Neutral",
    "Positive",
    "Negative",
    "Neutral",
    "Positive",
    "Negative",
    "Negative",
    "Positive",
]

# Now, we'll process each review using spaCy.
# spaCy tokenizes the text and adds linguistic annotations.
# We'll simulate a simple sentiment analysis based on the presence of certain keywords.

def get_sentiment(text):
    """
    A simple rule-based function to determine sentiment.
    In a more advanced model, you would train a classifier.
    """
    # Convert the text to lowercase for easier keyword matching.
    lower_text = text.lower()

    if "love" in lower_text or "superb" in lower_text or "masterpiece" in lower_text or "uplifting" in lower_text or "fantastic" in lower_text:

        return "Positive"

    elif "disappointment" in lower_text or "boring" in lower_text or "waste of time" in lower_text or "cringeworthy" in lower_text:

        return "Negative"

    else:

        return "Neutral"


# Let's see how our simple function performs.

print("--- Sentiment Analysis Results ---")

for i, review in enumerate(movie_reviews):

    predicted_sentiment = get_sentiment(review)

    actual_sentiment = sentiments[i]


    # Use spaCy to analyze the review

    doc = nlp(review)


    # Print the review, the predicted sentiment, and the actual sentiment for comparison.

    print(f"\nReview: '{review}'")

    print(f"Predicted Sentiment: {predicted_sentiment}")

    print(f"Actual Sentiment: {actual_sentiment}")


    # Let's break down the review to show the linguistic features spaCy gives us.

    print("--- Linguistic Analysis (spaCy) ---")

    for token in doc:

        # A token is a word or a punctuation mark.

        # token.text is the word itself.

        # token.pos_ is its part of speech.

        # token.dep_ is its grammatical dependency.

        print(f"Token: {token.text}\t Part of Speech: {token.pos_}\t Dependency: {token.dep_}")
```

```
# This is a very basic example. In real NLP projects, you would use machine learning models

# trained on massive datasets to achieve much more accurate results.
```

```
--- Sentiment Analysis Results ---


Review: 'I absolutely loved this film! The acting was superb and the plot was engaging.'

Predicted Sentiment: Positive

Actual Sentiment: Positive

--- Linguistic Analysis (spaCy) ---

Token: I Part of Speech: PRON    Dependency: nsubj

Token: absolutely          Part of Speech: ADV      Dependency: advmod

Token: loved      Part of Speech: VERB    Dependency: ROOT

Token: this       Part of Speech: DET     Dependency: det

Token: film       Part of Speech: NOUN   Dependency: dobj

Token: ! Part of Speech: PUNCT Dependency: punct

Token: The        Part of Speech: DET     Dependency: det

Token: acting     Part of Speech: NOUN   Dependency: nsubj

Token: was        Part of Speech: AUX     Dependency: ROOT

Token: superb     Part of Speech: NOUN   Dependency: attr

Token: and        Part of Speech: CCONJ Dependency: cc

Token: the        Part of Speech: DET     Dependency: det

Token: plot       Part of Speech: NOUN   Dependency: nsubj

Token: was        Part of Speech: AUX     Dependency: aux

Token: engaging            Part of Speech: VERB    Dependency: conj

Token: . Part of Speech: PUNCT Dependency: punct


Review: 'This movie was a total disappointment. It was boring and the ending made no sense.'

Predicted Sentiment: Negative

Actual Sentiment: Negative

--- Linguistic Analysis (spaCy) ---

Token: This       Part of Speech: DET     Dependency: det

Token: movie      Part of Speech: NOUN   Dependency: nsubj

Token: was        Part of Speech: AUX     Dependency: ROOT

Token: a          Part of Speech: DET     Dependency: det
```

```
Token: total        Part of Speech: ADJ      Dependency: amod
Token: disappointment   Part of Speech: NOUN   Dependency: attr
Token: . Part of Speech: PUNCT Dependency: punct
Token: It           Part of Speech: PRON     Dependency: nsubj
Token: was          Part of Speech: AUX      Dependency: ROOT
Token: boring       Part of Speech: ADJ      Dependency: acomp
Token: and          Part of Speech: CCONJ Dependency: cc
Token: the          Part of Speech: DET      Dependency: det
Token: ending       Part of Speech: NOUN     Dependency: nsubj
Token: made         Part of Speech: VERB     Dependency: conj
Token: no           Part of Speech: DET      Dependency: det
Token: sense        Part of Speech: NOUN     Dependency: dobj
Token: . Part of Speech: PUNCT Dependency: punct


Review: 'The film had its moments, but overall, it was just average.'
Predicted Sentiment: Neutral
Actual Sentiment: Neutral
--- Linguistic Analysis (spaCy) ---
Token: The          Part of Speech: DET      Dependency: det
Token: film         Part of Speech: NOUN     Dependency: nsubj
Token: had          Part of Speech: VERB     Dependency: ROOT
Token: its          Part of Speech: PRON     Dependency: poss
Token: moments            Part of Speech: NOUN   Dependency: dobj
Token: , Part of Speech: PUNCT Dependency: punct
Token: but          Part of Speech: CCONJ Dependency: cc
Token: overall      Part of Speech: ADV      Dependency: advmod
Token: , Part of Speech: PUNCT Dependency: punct
Token: it           Part of Speech: PRON     Dependency: nsubj
Token: was          Part of Speech: AUX      Dependency: conj
Token: just         Part of Speech: ADV      Dependency: advmod
Token: average   Part of Speech: ADJ      Dependency: acomp
```

```
Token: . Part of Speech: PUNCT Dependency: punct


Review: 'I was on the edge of my seat! What a thrilling experience!'

Predicted Sentiment: Neutral

Actual Sentiment: Positive

--- Linguistic Analysis (spaCy) ---

Token: I Part of Speech: PRON   Dependency: nsubj

Token: was        Part of Speech: AUX      Dependency: ROOT

Token: on         Part of Speech: ADP      Dependency: prep

Token: the        Part of Speech: DET      Dependency: det

Token: edge       Part of Speech: NOUN  Dependency: pobj

Token: of         Part of Speech: ADP      Dependency: prep

Token: my         Part of Speech: PRON   Dependency: poss

Token: seat       Part of Speech: NOUN  Dependency: pobj

Token: ! Part of Speech: PUNCT Dependency: punct

Token: What       Part of Speech: DET      Dependency: det

Token: a          Part of Speech: DET      Dependency: det

Token: thrilling  Part of Speech: NOUN  Dependency: compound

Token: experience          Part of Speech: NOUN   Dependency: ROOT

Token: ! Part of Speech: PUNCT Dependency: punct


Review: 'The plot was confusing and the characters were not believable. I would not recommend it.'

Predicted Sentiment: Neutral

Actual Sentiment: Negative

--- Linguistic Analysis (spaCy) ---

Token: The        Part of Speech: DET      Dependency: det

Token: plot       Part of Speech: NOUN  Dependency: nsubj

Token: was        Part of Speech: AUX      Dependency: ROOT

Token: confusing           Part of Speech: ADJ      Dependency: acomp

Token: and        Part of Speech: CCONJ Dependency: cc

Token: the        Part of Speech: DET      Dependency: det
```

```
Token: characters        Part of Speech: NOUN   Dependency: nsubj
Token: were      Part of Speech: AUX      Dependency: conj
Token: not       Part of Speech: PART     Dependency: neg
Token: believable        Part of Speech: ADJ      Dependency: acomp
Token: . Part of Speech: PUNCT Dependency: punct
Token: I Part of Speech: PRON    Dependency: nsubj
Token: would     Part of Speech: AUX      Dependency: aux
Token: not       Part of Speech: PART     Dependency: neg
Token: recommend         Part of Speech: VERB     Dependency: ROOT
Token: it        Part of Speech: PRON     Dependency: dobj
Token: . Part of Speech: PUNCT Dependency: punct

Review: 'The special effects were fantastic, but the story was weak.'
Predicted Sentiment: Positive
Actual Sentiment: Neutral
--- Linguistic Analysis (spaCy) ---
Token: The       Part of Speech: DET      Dependency: det
Token: special   Part of Speech: ADJ      Dependency: amod
Token: effects   Part of Speech: NOUN     Dependency: nsubj
Token: were      Part of Speech: AUX      Dependency: ROOT
Token: fantastic         Part of Speech: ADJ      Dependency: acomp
Token: , Part of Speech: PUNCT Dependency: punct
Token: but       Part of Speech: CCONJ Dependency: cc
Token: the       Part of Speech: DET      Dependency: det
Token: story     Part of Speech: NOUN     Dependency: nsubj
Token: was       Part of Speech: AUX      Dependency: conj
Token: weak      Part of Speech: ADJ      Dependency: acomp
Token: . Part of Speech: PUNCT Dependency: punct

Review: 'It's a cinematic masterpiece that will be remembered for years to come.'
Predicted Sentiment: Positive
```

```
Actual Sentiment: Positive

--- Linguistic Analysis (spaCy) ---

Token: It	Part of Speech: PRON	Dependency: nsubj

Token: 's	Part of Speech: AUX	Dependency: ROOT

Token: a	Part of Speech: DET	Dependency: det

Token: cinematic	Part of Speech: ADJ	Dependency: amod

Token: masterpiece	Part of Speech: NOUN	Dependency: attr

Token: that	Part of Speech: PRON	Dependency: nsubjpass

Token: will	Part of Speech: AUX	Dependency: aux

Token: be	Part of Speech: AUX	Dependency: auxpass

Token: remembered	Part of Speech: VERB	Dependency: relcl

Token: for	Part of Speech: ADP	Dependency: prep

Token: years	Part of Speech: NOUN	Dependency: pobj

Token: to	Part of Speech: PART	Dependency: aux

Token: come	Part of Speech: VERB	Dependency: relcl

Token: .	Part of Speech: PUNCT	Dependency: punct


Review: 'The acting was wooden and the dialogue was cringeworthy.'

Predicted Sentiment: Negative

Actual Sentiment: Negative

--- Linguistic Analysis (spaCy) ---

Token: The	Part of Speech: DET	Dependency: det

Token: acting	Part of Speech: NOUN	Dependency: nsubj

Token: was	Part of Speech: AUX	Dependency: ROOT

Token: wooden	Part of Speech: ADJ	Dependency: acomp

Token: and	Part of Speech: CCONJ	Dependency: cc

Token: the	Part of Speech: DET	Dependency: det

Token: dialogue	Part of Speech: NOUN	Dependency: nsubj

Token: was	Part of Speech: AUX	Dependency: conj

Token: cringeworthy	Part of Speech: ADJ	Dependency: acomp

Token: .	Part of Speech: PUNCT	Dependency: punct
```

```
Review: 'I have never been so bored in my life. A complete waste of time.'

Predicted Sentiment: Negative

Actual Sentiment: Negative

--- Linguistic Analysis (spaCy) ---

Token: I Part of Speech: PRON   Dependency: nsubj

Token: have       Part of Speech: AUX     Dependency: aux

Token: never      Part of Speech: ADV     Dependency: neg

Token: been       Part of Speech: AUX     Dependency: ROOT

Token: so         Part of Speech: ADV     Dependency: advmod

Token: bored      Part of Speech: ADJ     Dependency: acomp

Token: in         Part of Speech: ADP     Dependency: prep

Token: my         Part of Speech: PRON    Dependency: poss

Token: life       Part of Speech: NOUN    Dependency: pobj

Token: . Part of Speech: PUNCT Dependency: punct

Token: A          Part of Speech: DET     Dependency: det

Token: complete           Part of Speech: ADJ     Dependency: amod

Token: waste      Part of Speech: NOUN    Dependency: ROOT

Token: of         Part of Speech: ADP     Dependency: prep

Token: time       Part of Speech: NOUN    Dependency: pobj

Token: . Part of Speech: PUNCT Dependency: punct


Review: 'An uplifting and heartwarming story that everyone should see.'

Predicted Sentiment: Positive

Actual Sentiment: Positive

--- Linguistic Analysis (spaCy) ---

Token: An         Part of Speech: DET     Dependency: det

Token: uplifting  Part of Speech: ADJ     Dependency: amod

Token: and        Part of Speech: CCONJ Dependency: cc

Token: heartwarming       Part of Speech: NOUN    Dependency: conj

Token: story      Part of Speech: NOUN    Dependency: ROOT
```

```
Token: that       Part of Speech: PRON    Dependency: dobj
Token: everyone          Part of Speech: PRON    Dependency: nsubj
Token: should     Part of Speech: AUX     Dependency: aux
Token: see        Part of Speech: VERB    Dependency: relcl
Token: . Part of Speech: PUNCT Dependency: punct
```

---

---

---

```
# This script applies a more advanced rule-based sentiment analysis
# to a set of product and service reviews.


import spacy


# Load the small English language model for spaCy.
try:
  nlp = spacy.load("en_core_web_sm")
except OSError:
  print("The 'en_core_web_sm' model is not downloaded.")
  print("Please run 'python -m spacy download en_core_web_sm' in your terminal.")
  exit()


# We'll create a list of product/service reviews to analyze.
reviews = [
  "The coffee here is fantastic, and the service is always great!",
  "I was not impressed with the battery life of this phone. It's terrible.",
  "The software is okay, but the user interface is just average.",
  "This laptop is incredibly fast, a truly superb machine!",
  "The headphones are comfortable but the sound quality is poor.",
  "I have never been so disappointed with a delivery.",
  "The support team was quick and helpful, I would highly recommend them.",
  "The product did not meet my expectations.",
  "It's not a bad camera for the price.",
  "I'm very happy with my new headphones!",
]

# We'll manually classify each review for training and evaluation purposes.
sentiments = [
  "Positive",
  "Negative",
  "Neutral",
  "Positive",
  "Neutral",
  "Negative",
  "Positive",
  "Negative",
  "Positive", # Example of "not bad" being positive
  "Positive",
]

# Define positive, negative, and negation word lists.
positive_words = {"fantastic", "great", "okay", "superb", "helpful", "happy", "fast", "comfortable", "recommend"}
negative_words = {"disappointed", "terrible", "average", "poor", "unimpressed"}
negation_words = {"not", "no", "never", "n't", "didnt", "don't"}

def get_sentiment(text):
  """
  A more advanced rule-based function to determine sentiment,
  including a check for negation words.
  """
  words = text.lower().replace("'", "").split()
  sentiment_score = 0

  # Iterate through the words with their indices.
```

```
  for i, word in enumerate(words):
    if word in positive_words:
      # Check the previous two words for a negation.
      if i > 0 and words[i-1] in negation_words or (i > 1 and words[i-2] in negation_words):
        sentiment_score -= 1 # Flip to negative if a negation is found
      else:
        sentiment_score += 1 # Add a point for a positive word
    elif word in negative_words:
      if i > 0 and words[i-1] in negation_words or (i > 1 and words[i-2] in negation_words):
        sentiment_score += 1 # Flip to positive
      else:
        sentiment_score -= 1 # Subtract a point for a negative word

  if sentiment_score > 0:
    return "Positive"
  elif sentiment_score < 0:
    return "Negative"
  else:
    return "Neutral"

# Let's see how our more advanced function performs.
print("--- Sentiment Analysis Results ---")
for i, review in enumerate(reviews):
  predicted_sentiment = get_sentiment(review)
  actual_sentiment = sentiments[i]

  # Use spaCy to analyze the review
  doc = nlp(review)

  # Print the review, the predicted sentiment, and the actual sentiment for comparison.
  print(f"\nReview: '{review}'")
  print(f"Predicted Sentiment: {predicted_sentiment}")

  print(f"Actual Sentiment: {actual_sentiment}")


  # Let's break down the review to show the linguistic features spaCy gives us.

  print("--- Linguistic Analysis (spaCy) ---")

  for token in doc:

    print(f"Token: {token.text}\t Part of Speech: {token.pos_}\t Dependency: {token.dep_}")
```