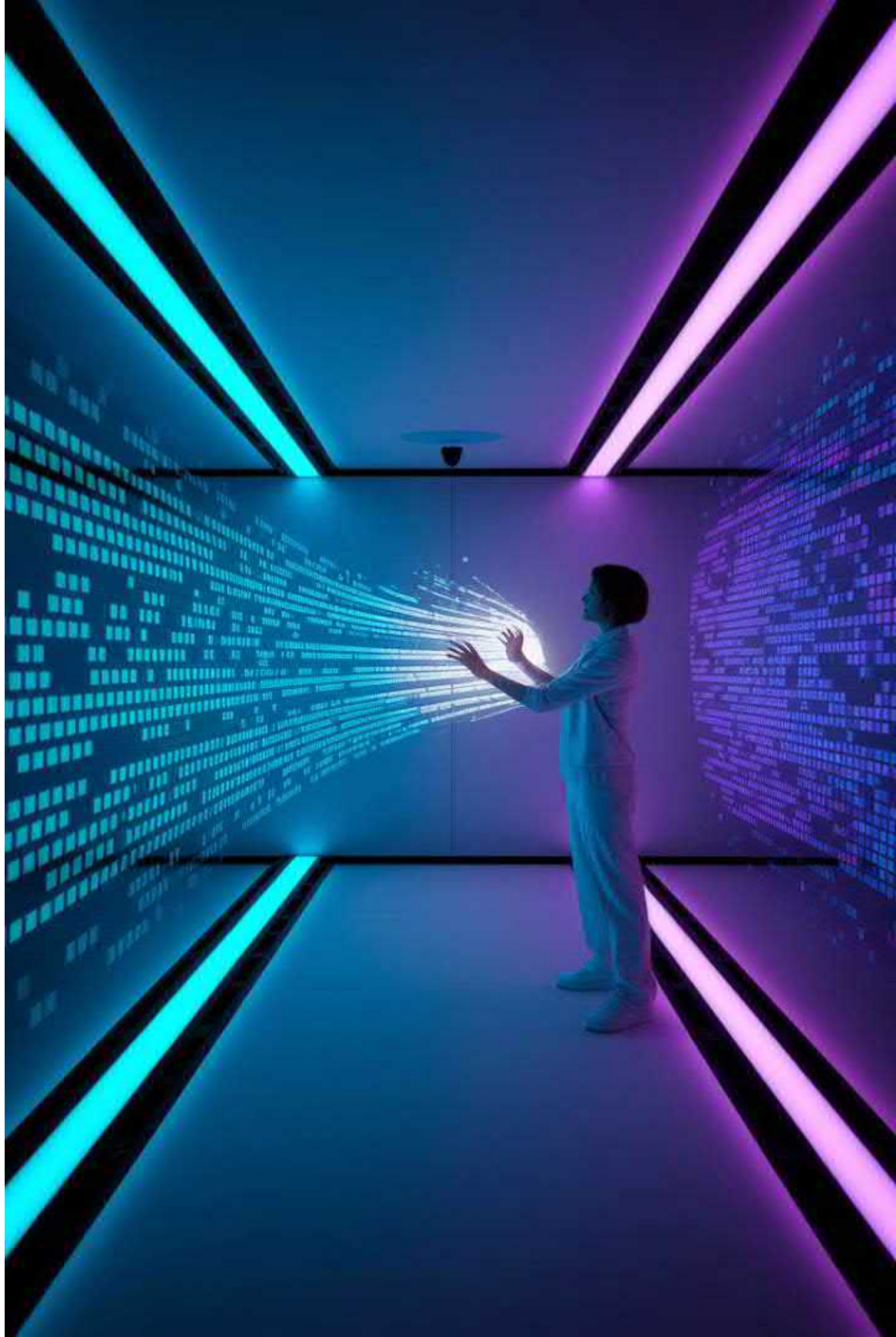
Transforming Text Analysis with NLP and Generative AI: From Fundamentals to Advanced Techniques

Explore the powerful intersection of NLP and Generative AI technologies. Discover practical applications and insights driving innovation.

Yong-Bin Kang
Senior Data Science Research Fellow
Swinburne University of Technology



Presentation Agenda

1

Introduction to NLP & Generative AI

Core concepts, historical development, and the impact of language technologies.

2

NLP Use Cases

Real-world NLP value and innovation.

3

NLP Foundations

Fundamental NLP methods.

4

Overview of Generative AI

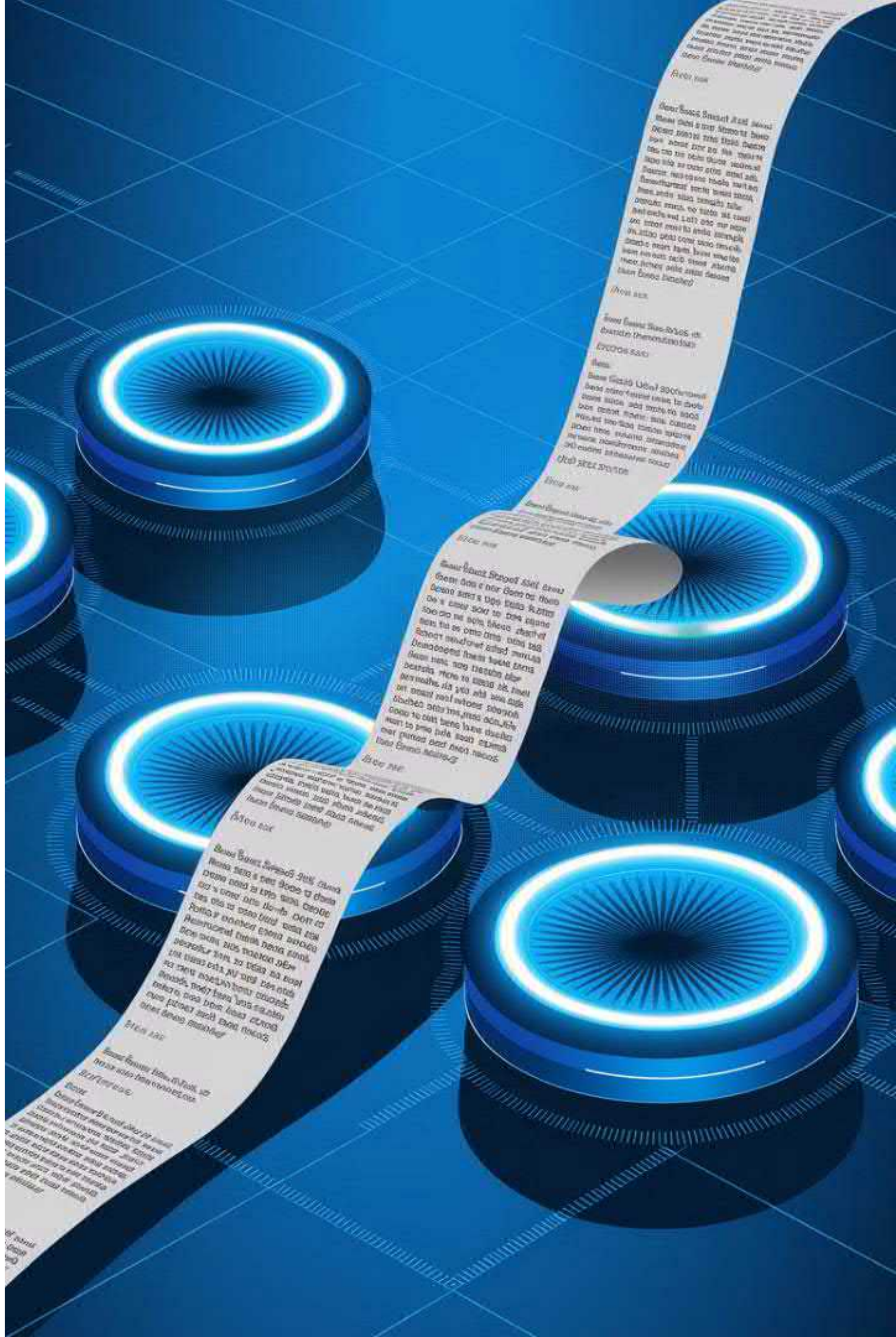
Generative AI systems to create human-like content.

5

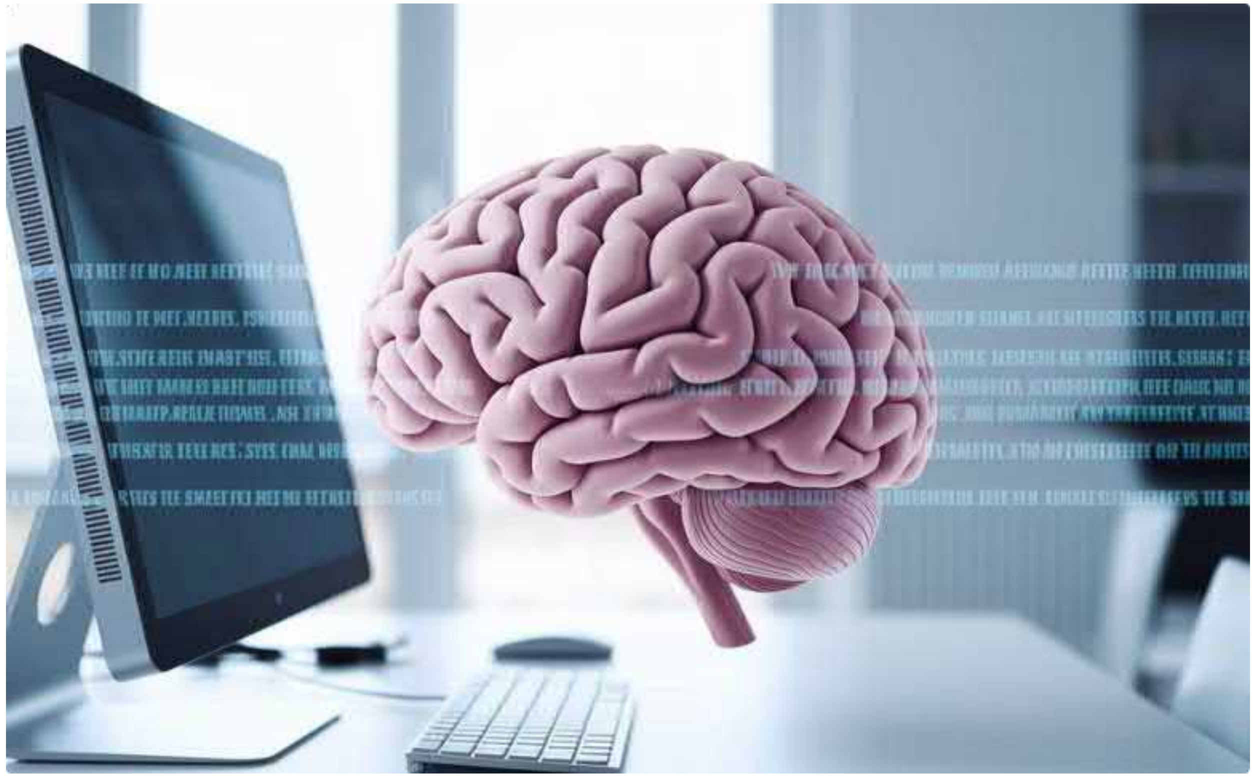
Conclusion

Introduction to NLP & Generative AI

Exploring the fundamentals of Natural Language Processing and Generative AI Techniques.



What is Natural Language Processing (NLP)?



Definition

NLP is a branch of AI that enables computers to understand, interpret, and interpret, and generate human language through computational algorithms. It algorithms. It bridges the gap between linguistic structure and machine machine learning to process unstructured text data at scale.

Applications

Span diverse fields: syntactic & semantic analysis, sentiment analysis, text analysis, text classification, machine translation, chatbot development, text development, text summarisation, and many more, enabling computers to computers to understand and interact with human language.

Generative AI

Definition

A type of AI that create new content, content, capable of generating human-indistinguishable outputs. outputs.

Capabilities

Generates sophisticated text (essays, code, poetry, etc), photorealistic images, music compositions, video sequences, and conversational responses with contextual understanding.

Foundation

Powered by Transformer-based architectures with trillions of parameters trained on diverse datasets (GPT-4: 1.8 trillion parameters trained with around 10 trillion tokens, requiring 3 months of training using 8000 H100 GPUs), enabling incredibly complex pattern recognition and synthesis.

Revolution

Enable creative problem-solving, solving, content personalisation, and and autonomous system development.

Evolution of Language Technologies

The development of language technologies has progressed through distinct phases, each building upon previous innovations. innovations.

Rule-Based NLP (1950s- (1950s-1990s)

Early systems like ELIZA (1966) (1966) and SHRDLU (1970s) relied relied on hard-coded grammatical grammatical rules, pattern matching, and symbolic logic. logic.

Statistical NLP (1990s-2010s)

Hidden Markov Models and Support Vector Machines enabled enabled probability-based approaches.

Neural NLP (2010s-2018)

Word2Vec (2013) and recurrent recurrent neural networks (LSTMs/GRUs) enabled contextual contextual word representations. representations.

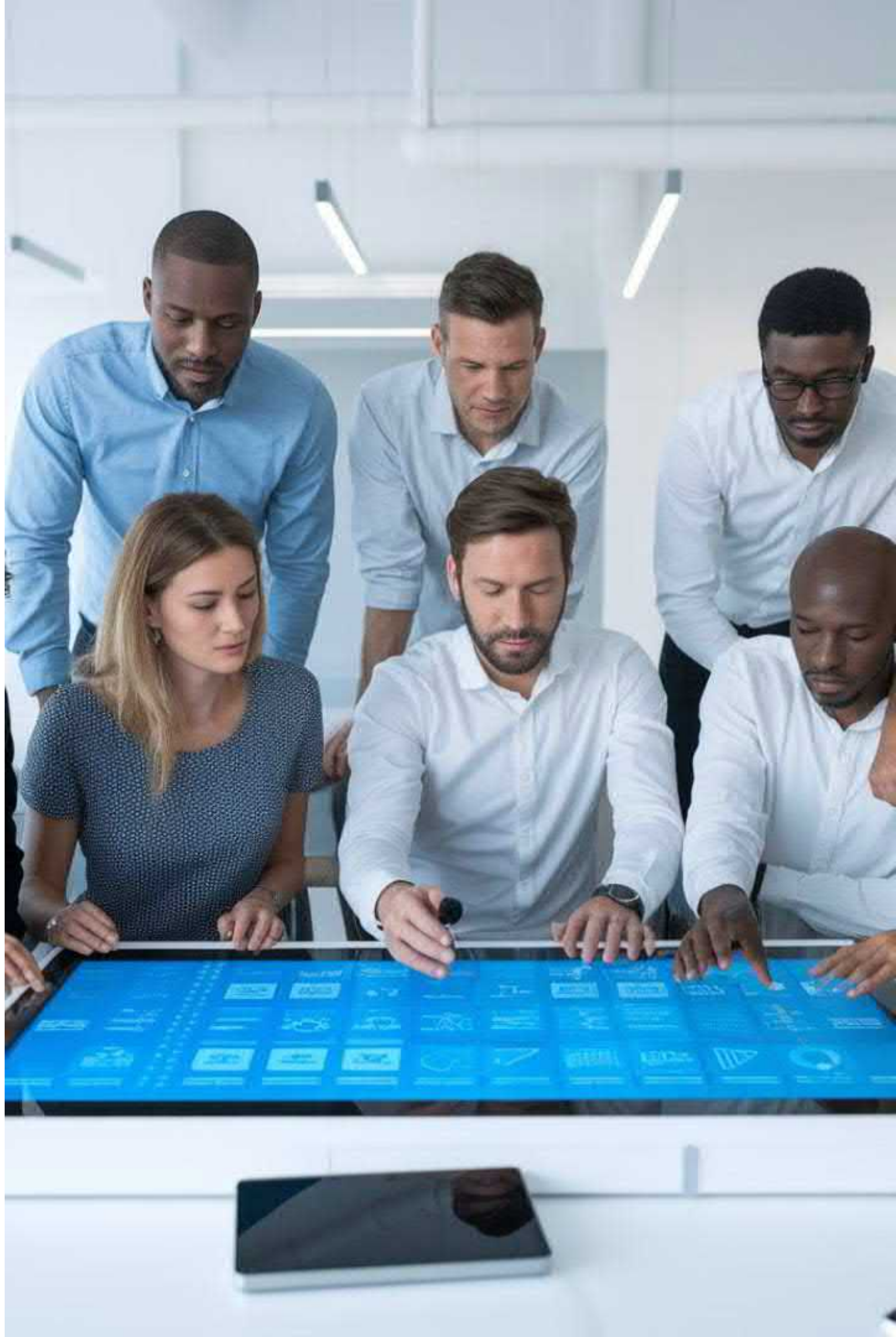
Generative AI (2018-Present)

Transformer architecture (2017) enabled parallel processing and self-attention mechanisms.

GPT series (OpenAI), BERT (Google), (Google), and subsequent models models revolutionise language understanding and generation, achieving near-human performance. performance.

NLP Use Cases

NLP helps solve many everyday problems, making technology more accessible and useful.



NLP Applications: Sentiment Analysis

NLP has the ability to understand the emotional tone and sentiment expressed within written content.

Sentiment Analysis in Legal Documents

A study compared Pre-Sentence Reports (PSRs) between Indigenous sentencing courts and mainstream courts in Australia based on their sentence expressions in PSRs.

Risk vs. Strength Language

We found language about "risks" was more prevalent than discussions of personal strengths or cultural factors in PSRs in the mainstream courts.

Differences

PSRs in Indigenous sentencing courts showed more positive language than negative language.
This study highlights the importance of considering **cultural factors** when sentencing Indigenous people.

Culture, Strengths, and Risk: The Language of Pre-Sentence Reports in Indigenous Sentencing Courts and Mainstream Courts (Darcy, Forkan, Yong-Bin et al., 2022)

NLP Applications: Financial Advice Audit

NLP can analyse complex financial advice documents, which are often dense with regulatory and technical language, making manual review time-consuming and challenging.

Manual SoA Audit Challenge

Quality in personal financial advice (Statements of Advice: SoA) is important but hard to check. Manual review takes too much time and tend to be subjective, limiting how many documents can be checked.

NLP Solution

We developed an Q&A system that can automatically audit the quality of a SoA document and estimate its risk levels.

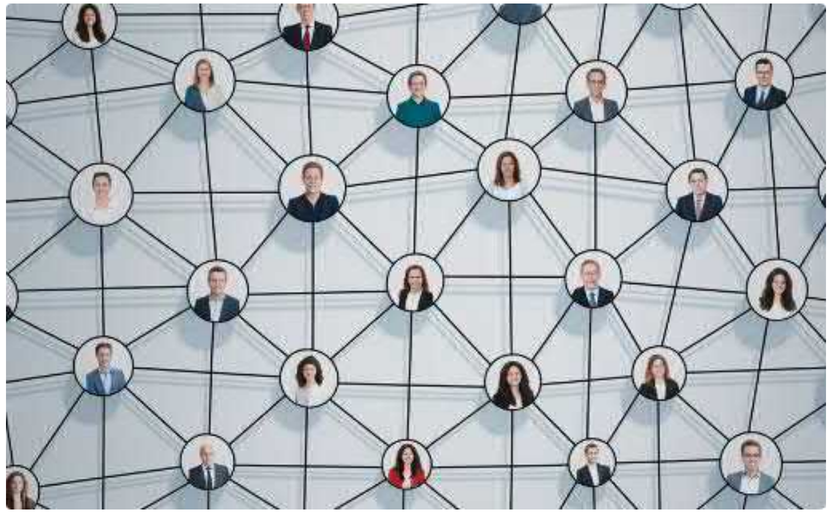
Transforming Financial Compliance

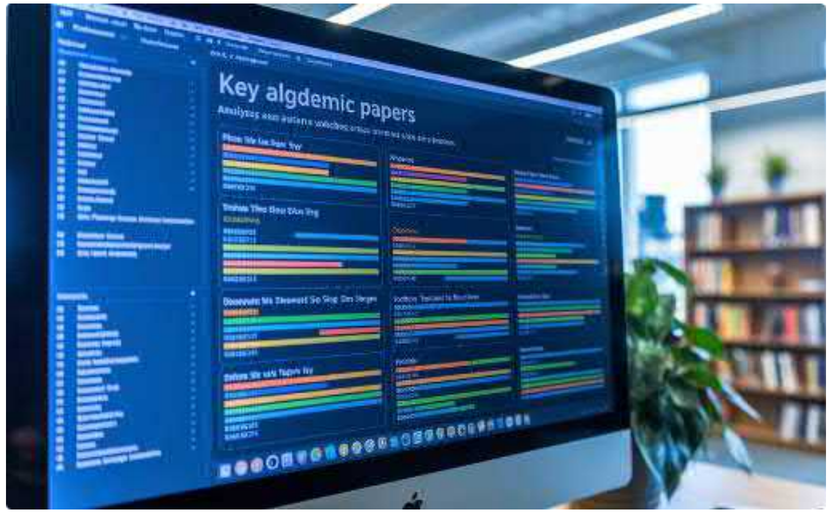
This tool shows how NLP can transform compliance checking and finding problems more accurately across financial services.

SRAuditor: An Automated Assessment Tool for Statement of Advice Documents (Kang et al., 2022)

NLP Applications: Expert Finding

NLP can analyse large volumes of text-heavy data, such as academic publications, to identify experts in specific fields.





Text Analysis for Expertise

NLP analyses academic publications, patents, and online content to identify leading researchers in specialised fields.

Expertise Mapping

We developed NLP algorithms to identify experts across various fields by thoroughly analysing their research topics, their usage patterns, and professional networks.

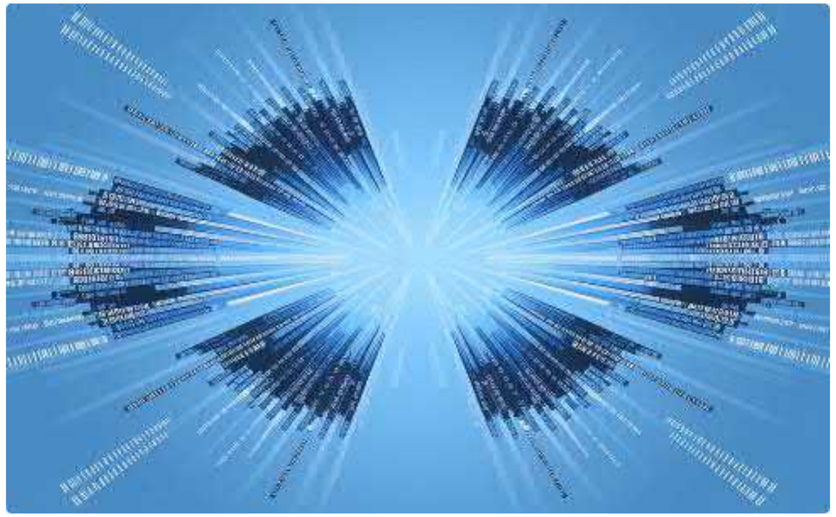
Facilitating Collaboration

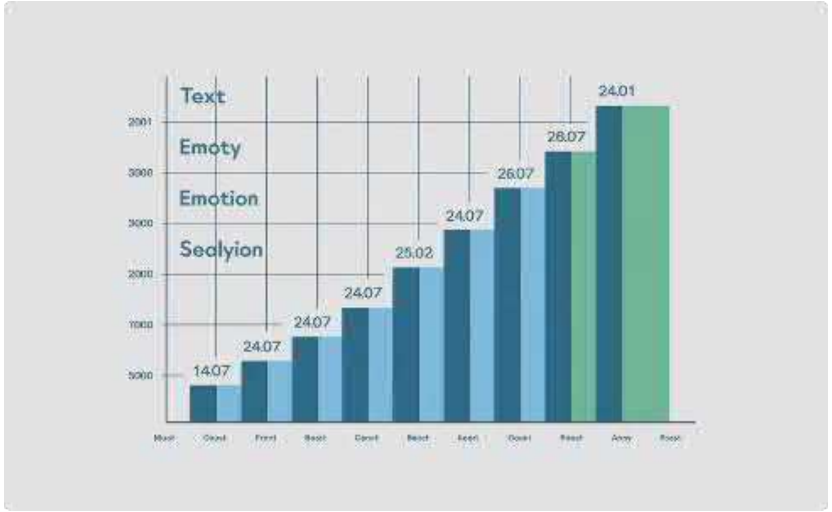
This expert identification system can be used to accurately understand subject matter experts, enhancing knowledge sharing and research partnerships.

ExpFinder: A hybrid model for expert finding from text-based expertise data (Kang et al., 2023)

NLP Applications: Online Mental Health Forum Analysis

NLP can analyse online mental health discussions by processing large volumes of user-generated content to detect emotional patterns, identify key themes, and assess language indicative of psychological distress or resilience.





Topic Modeling in Mental Health

NLP techniques identify key themes and patterns in online mental health forums, revealing insights about community needs and concerns.

Emotional Pattern Detection

Topic modelling detected emotional patterns and language indicative of psychological distress or resilience in user-generated content.

Resilience Identification

The identified resilience factors can foster community support, contributing to effective mental health interventions.

Resilience in Web-Based Mental Health Communities: Building a Resilience Dictionary With Semi-automatic Text Analysis (Kang et al., 2022).

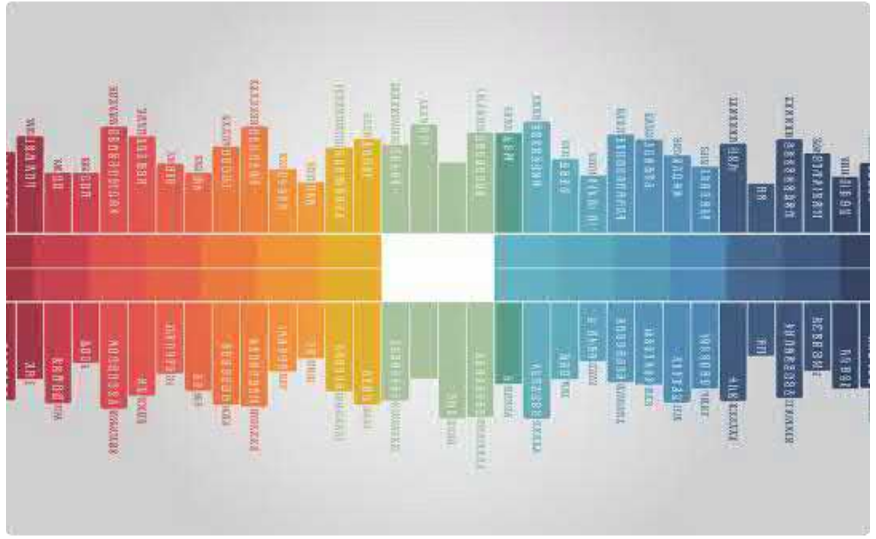
Leveraging stylometry analysis to identify unique characteristics of peer support user groups in online mental health forums (Kang et al., 2023)

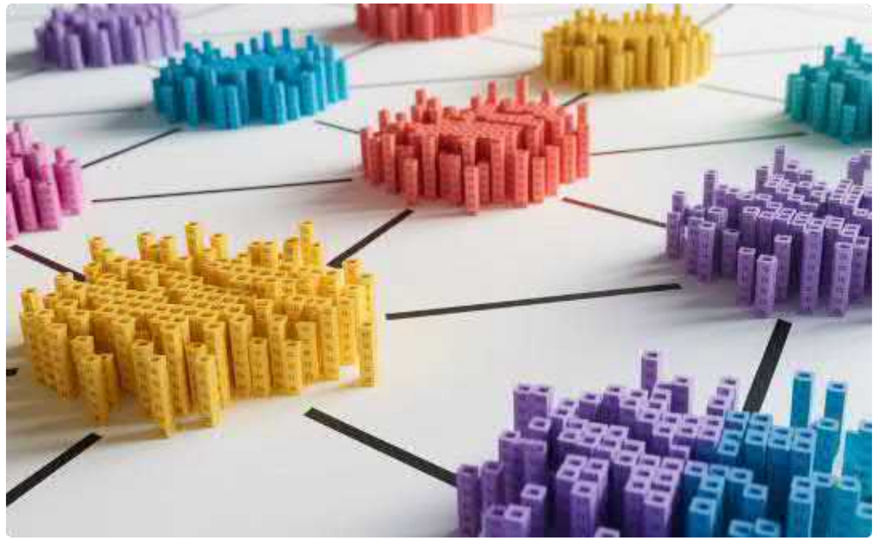
NLP Foundations

Fundamental NLP methods.

NLP Foundation

We will explore these NLP foundation techniques today!





Syntactic/Semantic Analysis

Analyse syntactic structures from text & identify semantics of words

Sentiment/Emotion Analysis

Identify sentiments of words and Sentences and emotional tone from text

Topic Modeling

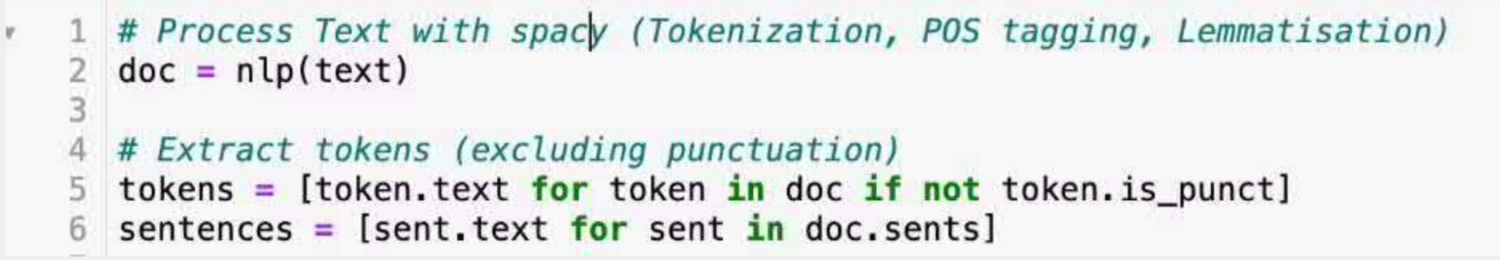
Identify key themes (or topics) in text

Syntactic Structure Analysis (1)

Syntactic analysis breaks down text into its basic components to understand its structure. We use the "**spacy**" NLP package: a powerful and efficient library for NLP efficient library for NLP in Python.

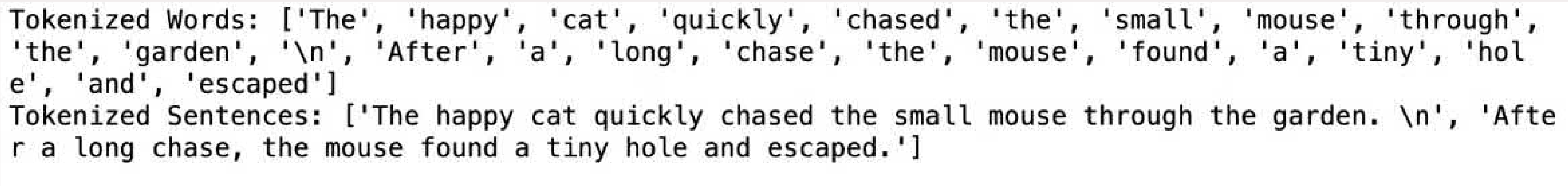
Text: *"The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

Tokenisation: Breaking text into individual sentences and words



```
# Process Text with spacy (Tokenization, POS tagging, Lemmatisation)
doc = nlp(text)

# Extract tokens (excluding punctuation)
tokens = [token.text for token in doc if not token.is_punct]
sentences = [sent.text for sent in doc.sents]
```

Result



```
Tokenized Words: ['The', 'happy', 'cat', 'quickly', 'chased', 'the', 'small', 'mouse', 'through',
'the', 'garden', '\n', 'After', 'a', 'long', 'chase', 'the', 'mouse', 'found', 'a', 'tiny', 'hol
e', 'and', 'escaped']
Tokenized Sentences: ['The happy cat quickly chased the small mouse through the garden. \n', 'Afte
r a long chase, the mouse found a tiny hole and escaped.']
```

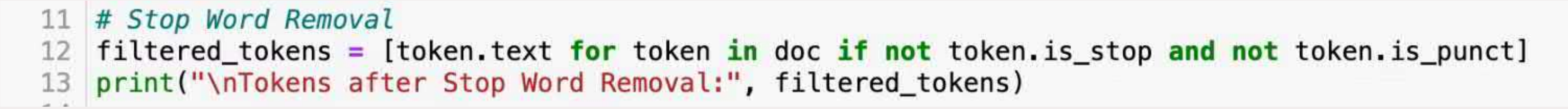
◆ **Key Takeaways: Tokenisation** splits text into words/sentences for further processing.

Syntactic Structure Analysis (2)

Text: *"The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

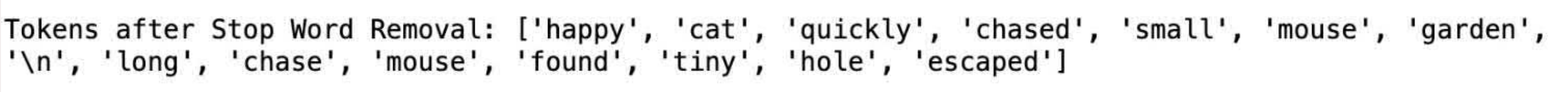
escaped."

Stopwords Removal: Eliminate unimportant words



```python
# Stop Word Removal
filtered_tokens = [token.text for token in doc if not token.is_stop and not token.is_punct]
print("\nTokens after Stop Word Removal:", filtered_tokens)
```

Result



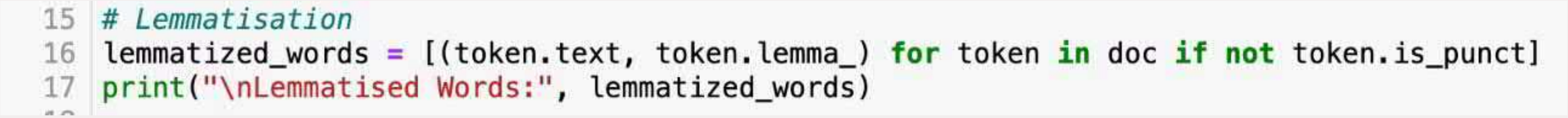
```
Tokens after Stop Word Removal: ['happy', 'cat', 'quickly', 'chased', 'small', 'mouse', 'garden',
'\n', 'long', 'chase', 'mouse', 'found', 'tiny', 'hole', 'escaped']
```

◆ **Key Takeaways: Stop Word Removal** eliminates unimportant words, improving text analysis efficiency.

Syntactic Structure Analysis (3)

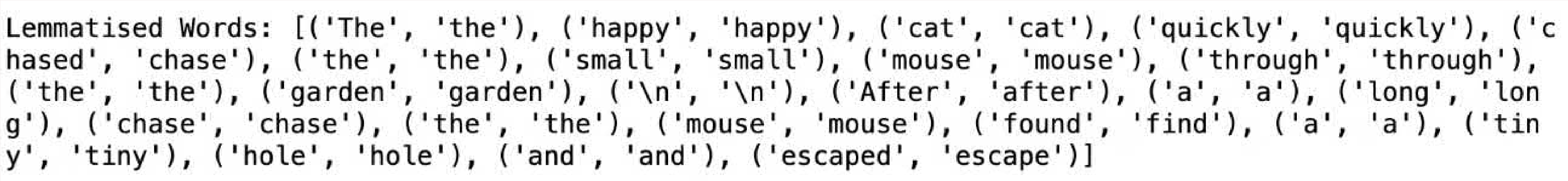
Text: *"The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

Lemmatisation: Reduce words to their base form or dictionary form (lemma) while considering the word’s context and Part-of-Speech (POS).



```python
# Lemmatisation
lemmatized_words = [(token.text, token.lemma_) for token in doc if not token.is_punct]
print("\nLemmatised Words:", lemmatized_words)
```

Result



```
Lemmatised Words: [('The', 'the'), ('happy', 'happy'), ('cat', 'cat'), ('quickly', 'quickly'), ('c
hased', 'chase'), ('the', 'the'), ('small', 'small'), ('mouse', 'mouse'), ('through', 'through'),
('the', 'the'), ('garden', 'garden'), ('\n', '\n'), ('After', 'after'), ('a', 'a'), ('long', 'lon
g'), ('chase', 'chase'), ('the', 'the'), ('mouse', 'mouse'), ('found', 'find'), ('a', 'a'), ('tin
y', 'tiny'), ('hole', 'hole'), ('and', 'and'), ('escaped', 'escape')]
```

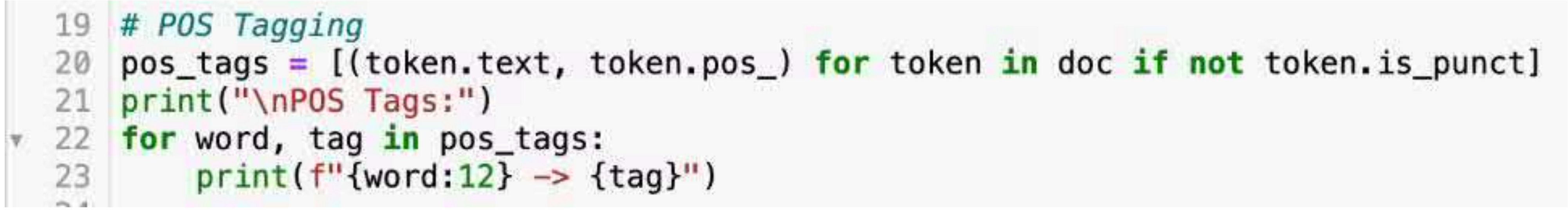
◆ **Key Takeaways:**

1) Standardise words for better text analysis (e.g., "running", "runs", and "ran" as the same word: "run")
2) Improve information retrieval (e.g., "jumping" should also return results for "jump" or "jumped")

Syntactic Structure Analysis (4)

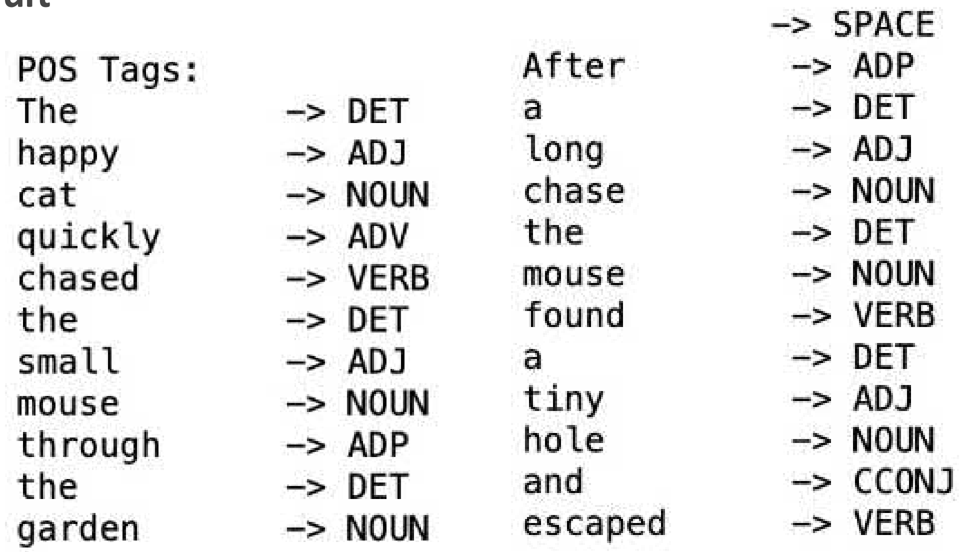
Text: "The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."

POS Tagging: POS is the process of assigning grammatical categories (such as noun, verb, adjective) to each adjective) to each word in a sentence. It helps NLP models understand the **role and function** of words in words in context.



```
# POS Tagging
pos_tags = [(token.text, token.pos_) for token in doc if not token.is_punct]
print("\nPOS Tags:")
for word, tag in pos_tags:
    print(f"{word:12} -> {tag}")
```

Result



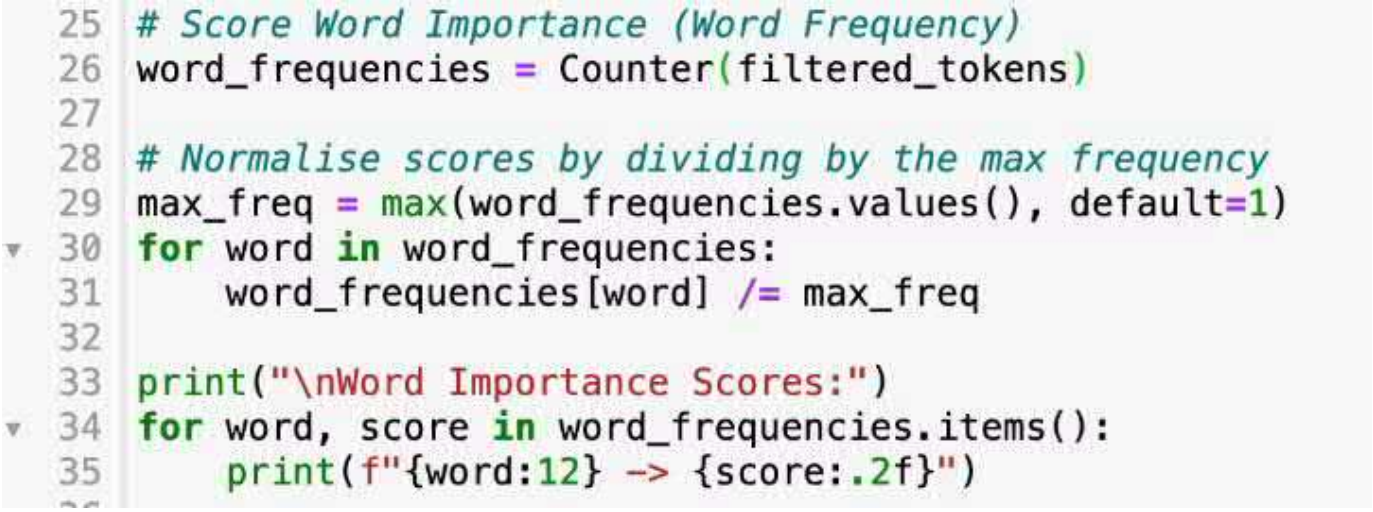
```
POS Tags:
The          -> DET
happy        -> ADJ
cat          -> NOUN
quickly      -> ADV
chased       -> VERB
the          -> DET
small        -> ADJ
mouse        -> NOUN
through      -> ADP
the          -> DET
garden       -> NOUN
             -> SPACE
After        -> ADP
a            -> DET
long         -> ADJ
chase        -> NOUN
the          -> DET
mouse        -> NOUN
found        -> VERB
a            -> DET
tiny         -> ADJ
hole         -> NOUN
and          -> CCONJ
escaped      -> VERB
```

◆ **Key Takeaways:** POS tagging helps machines **interpret language structure, improve comprehension (how words function together in context)**

Syntactic Structure Analysis (5)

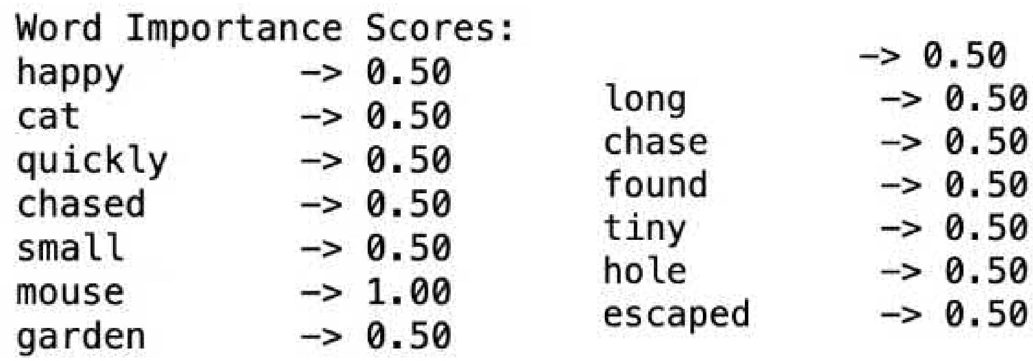
Text: *"The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

Measure word Importance: Determine how **significant** a word is in text (the following is a very simple simple approach)



```python
# Score Word Importance (Word Frequency)
word_frequencies = Counter(filtered_tokens)

# Normalise scores by dividing by the max frequency
max_freq = max(word_frequencies.values(), default=1)
for word in word_frequencies:
    word_frequencies[word] /= max_freq

print("\nWord Importance Scores:")
for word, score in word_frequencies.items():
    print(f"{word:12} -> {score:.2f}")
```

Result



```
Word Importance Scores:
happy        -> 0.50
cat          -> 0.50
quickly      -> 0.50
chased       -> 0.50
small        -> 0.50
mouse        -> 1.00
garden       -> 0.50
```

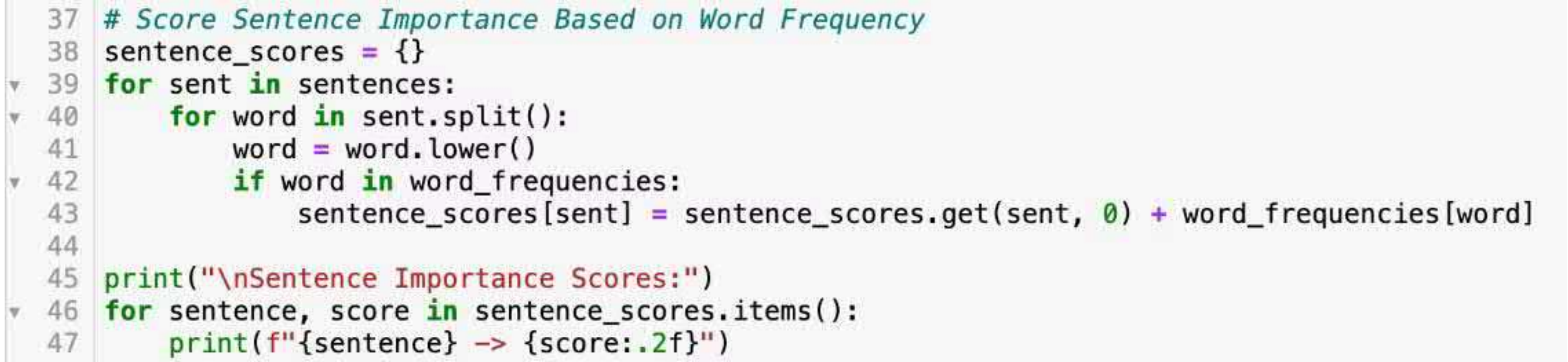
```
             -> 0.50
long         -> 0.50
chase        -> 0.50
found        -> 0.50
tiny         -> 0.50
hole         -> 0.50
escaped      -> 0.50
```

🔷 **Key Takeaways:** Helps extract **meaningful words** and estimate **main ideas** from text

Syntactic Structure Analysis (6)

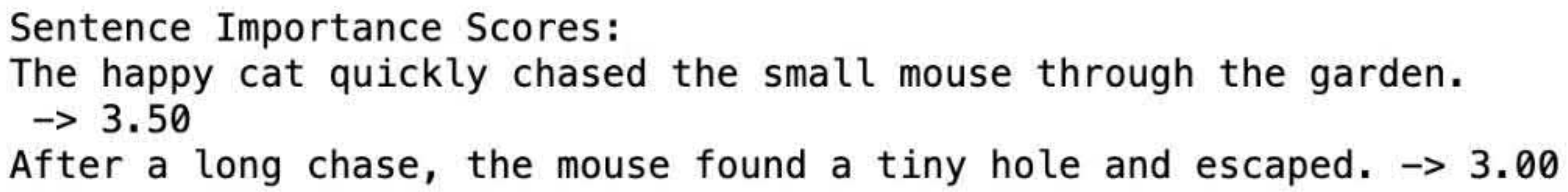
Text: *"The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

Measure sentence Importance: Determine which sentences **carry the most valuable information** in a text. **information** in a text. It is commonly used in **text summarisation**



```python
# Score Sentence Importance Based on Word Frequency
sentence_scores = {}
for sent in sentences:
    for word in sent.split():
        word = word.lower()
        if word in word_frequencies:
            sentence_scores[sent] = sentence_scores.get(sent, 0) + word_frequencies[word]

print("\nSentence Importance Scores:")
for sentence, score in sentence_scores.items():
    print(f"{sentence} -> {score:.2f}")
```

Result



```
Sentence Importance Scores:
The happy cat quickly chased the small mouse through the garden.
 -> 3.50
After a long chase, the mouse found a tiny hole and escaped. -> 3.00
```

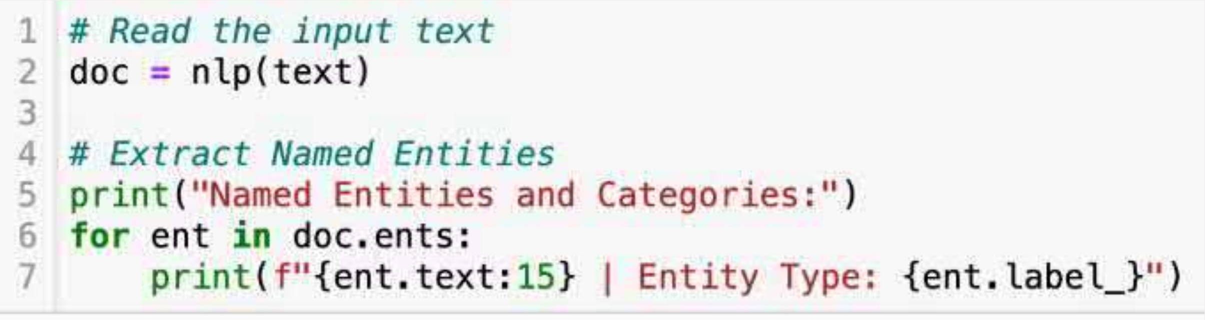
◆ **Key Takeaways:** Help to **identify the most informative sentences** and remove redundant or less valuable content

Semantic Analysis (1)

Semantic analysis is the process of understanding the **meaning** and **context** of words, phrases, and sentences in text. It focuses on **how words relate to each other**, their **intended meaning**, and **the concepts they represent**.

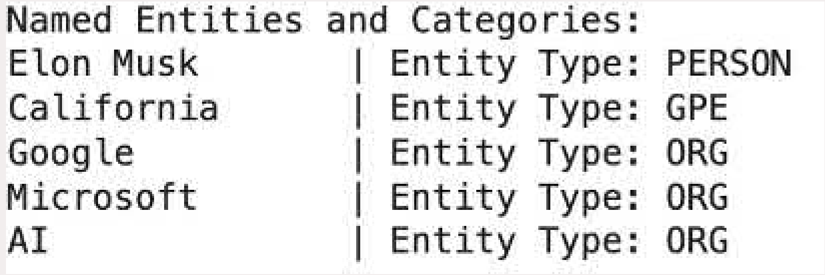
Text: *"Elon Musk founded SpaceX in California. Google and Microsoft are competing in the AI race."*

Named Entity Recognition (NER) – Extracting Key Entities: Identify **real-world objects** like people, places, and organisations.



```python
# Read the input text
doc = nlp(text)

# Extract Named Entities
print("Named Entities and Categories:")
for ent in doc.ents:
    print(f"{ent.text:15} | Entity Type: {ent.label_}")
```

Result



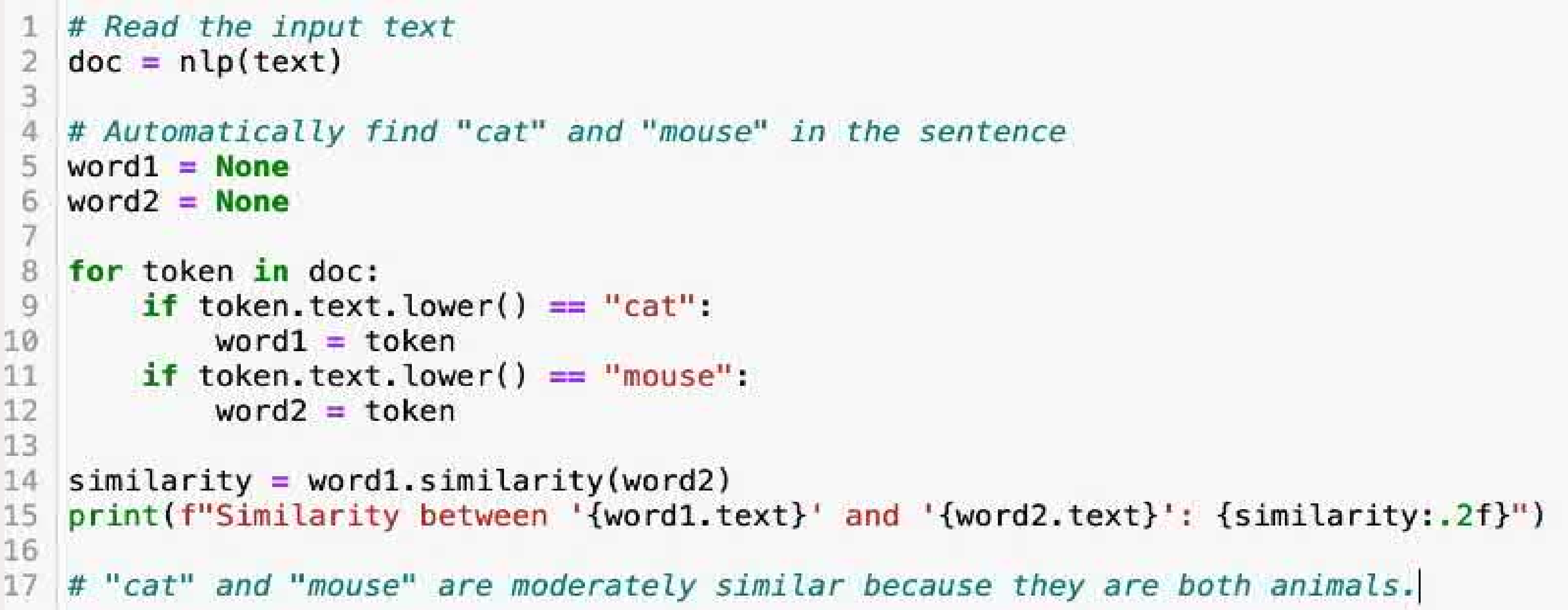
```
Named Entities and Categories:
Elon Musk       | Entity Type: PERSON
California      | Entity Type: GPE
Google          | Entity Type: ORG
Microsoft       | Entity Type: ORG
AI              | Entity Type: ORG
```

◆ **Key Takeaways:** **NER** can be used to identify key information pieces from text.

Semantic Analysis (2)

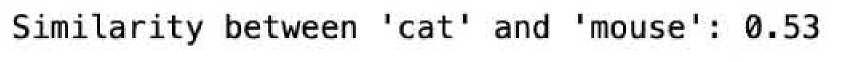
Text: *"The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

Measuring Word Similarity: Compare words based on their semantic meaning using word embeddings (e.g., Word2Vec). Word embedding can convert words into numerical vectors, capturing their meanings and relationships with other words in context.



```python
# Read the input text
doc = nlp(text)

# Automatically find "cat" and "mouse" in the sentence
word1 = None
word2 = None

for token in doc:
    if token.text.lower() == "cat":
        word1 = token
    if token.text.lower() == "mouse":
        word2 = token

similarity = word1.similarity(word2)
print(f"Similarity between '{word1.text}' and '{word2.text}': {similarity:.2f}")

# "cat" and "mouse" are moderately similar because they are both animals.
```

Result



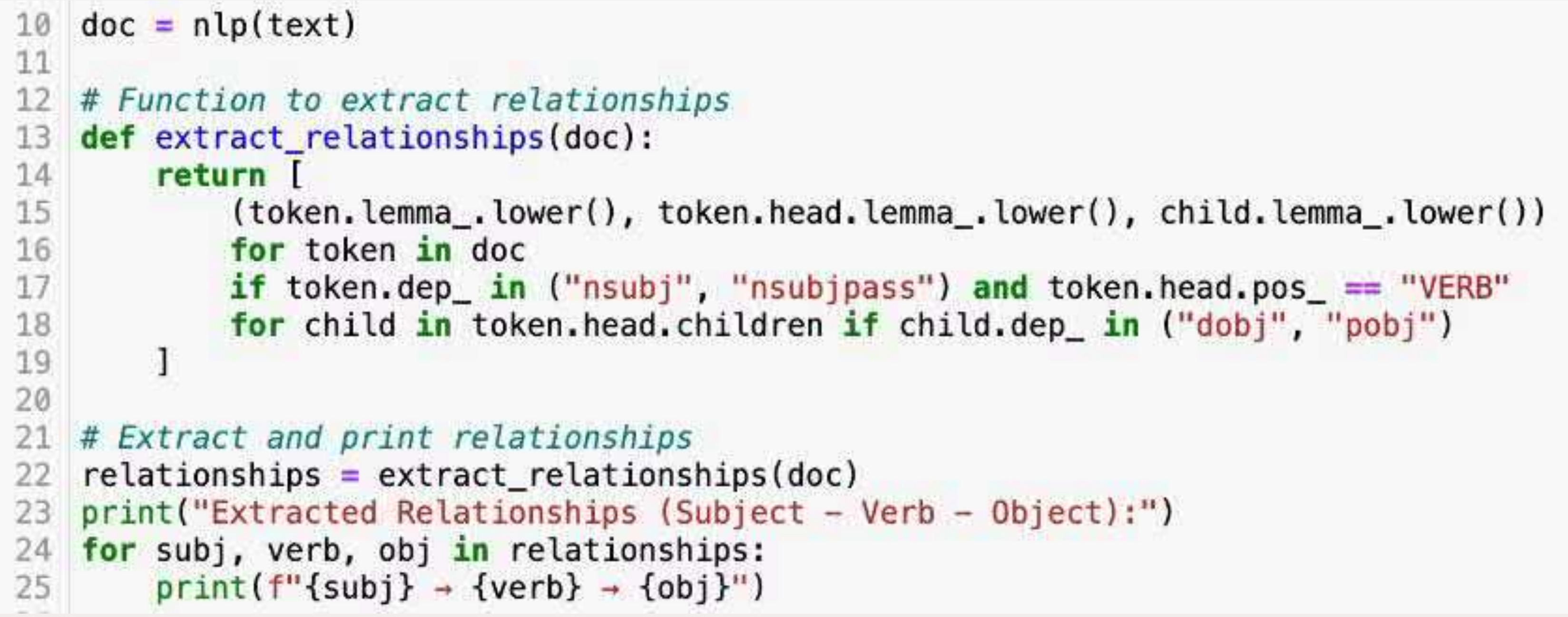
```
Similarity between 'cat' and 'mouse': 0.53
```

◆ Key Takeaways: Word embeddings help the identification of conceptual relationships of words.

Semantic Analysis (3)

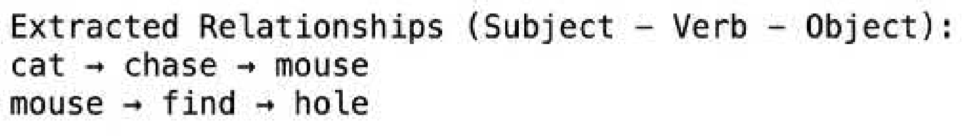
Text: *”The happy cat quickly chased the small mouse through the garden. After a long chase, the mouse found a tiny hole and escaped."*

Relationship Extraction: Identifying semantic relationships between words in a sentence.



```python
doc = nlp(text)

# Function to extract relationships
def extract_relationships(doc):
    return [
        (token.lemma_.lower(), token.head.lemma_.lower(), child.lemma_.lower())
        for token in doc
        if token.dep_ in ("nsubj", "nsubjpass") and token.head.pos_ == "VERB"
        for child in token.head.children if child.dep_ in ("dobj", "pobj")
    ]

# Extract and print relationships
relationships = extract_relationships(doc)
print("Extracted Relationships (Subject - Verb - Object):")
for subj, verb, obj in relationships:
    print(f"{subj} → {verb} → {obj}")
```

Result



```
Extracted Relationships (Subject - Verb - Object):
cat → chase → mouse
mouse → find → hole
```

◆ **Key Takeaways:** Improves Search & Retrieval for Answers:

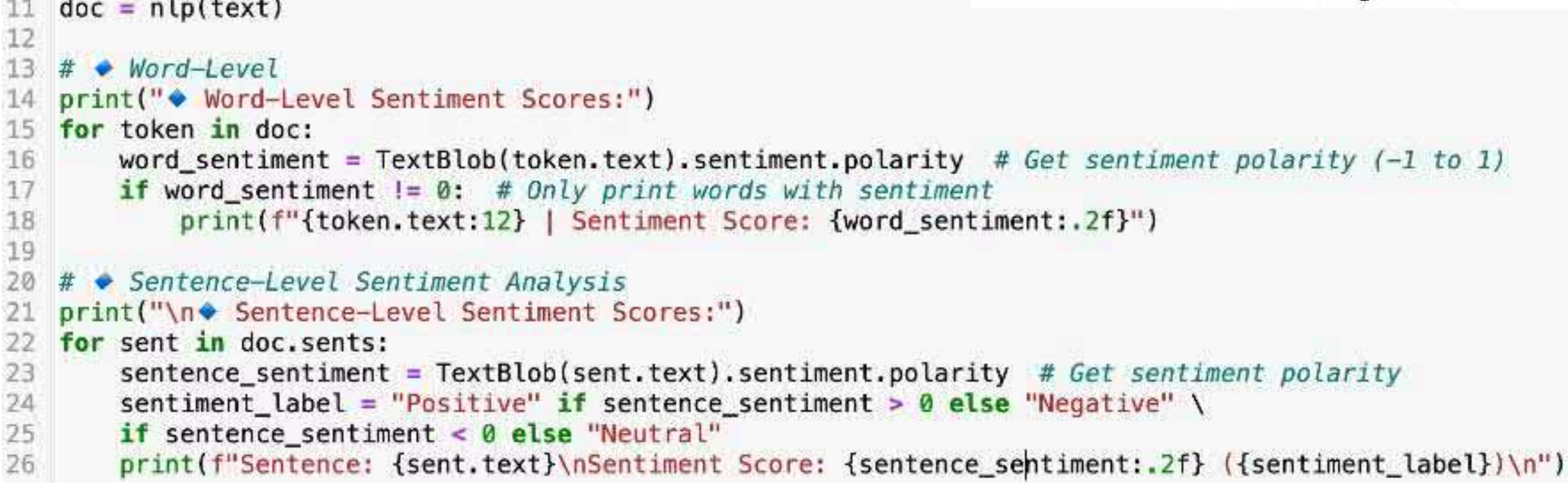
- Question: "Who chased mouse?" - With Relationship Extraction: AI quickly finds: "cat → chased → mouse".

Sentiment Analysis (1)

Sentiment analysis (also known as opinion mining) is the process of determining the emotional tone expressed in text.

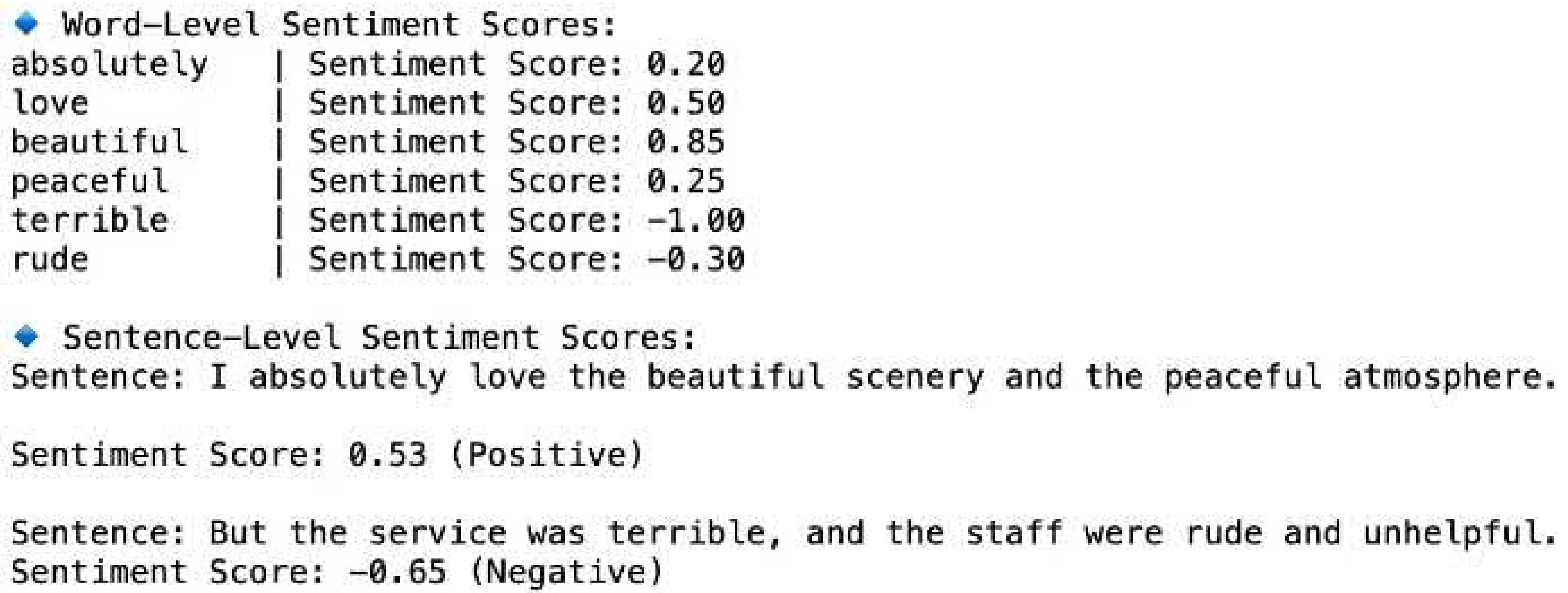
Text: *"I absolutely love the beautiful scenery and the peaceful atmosphere. But the service was terrible, and the staff were rude and rude and unhelpful."*

Word & Sentence-level Sentiment Analysis: Identify the binary emotions of words and sentences



```python
doc = nlp(text)

# ◆ Word-Level
print("◆ Word-Level Sentiment Scores:")
for token in doc:
    word_sentiment = TextBlob(token.text).sentiment.polarity  # Get sentiment polarity (-1 to 1)
    if word_sentiment != 0:  # Only print words with sentiment
        print(f"{token.text:12} | Sentiment Score: {word_sentiment:.2f}")

# ◆ Sentence-Level Sentiment Analysis
print("\n◆ Sentence-Level Sentiment Scores:")
for sent in doc.sents:
    sentence_sentiment = TextBlob(sent.text).sentiment.polarity  # Get sentiment polarity
    sentiment_label = "Positive" if sentence_sentiment > 0 else "Negative" \
    if sentence_sentiment < 0 else "Neutral"
    print(f"Sentence: {sent.text}\nSentiment Score: {sentence_sentiment:.2f} ({sentiment_label})\n")
```

Result

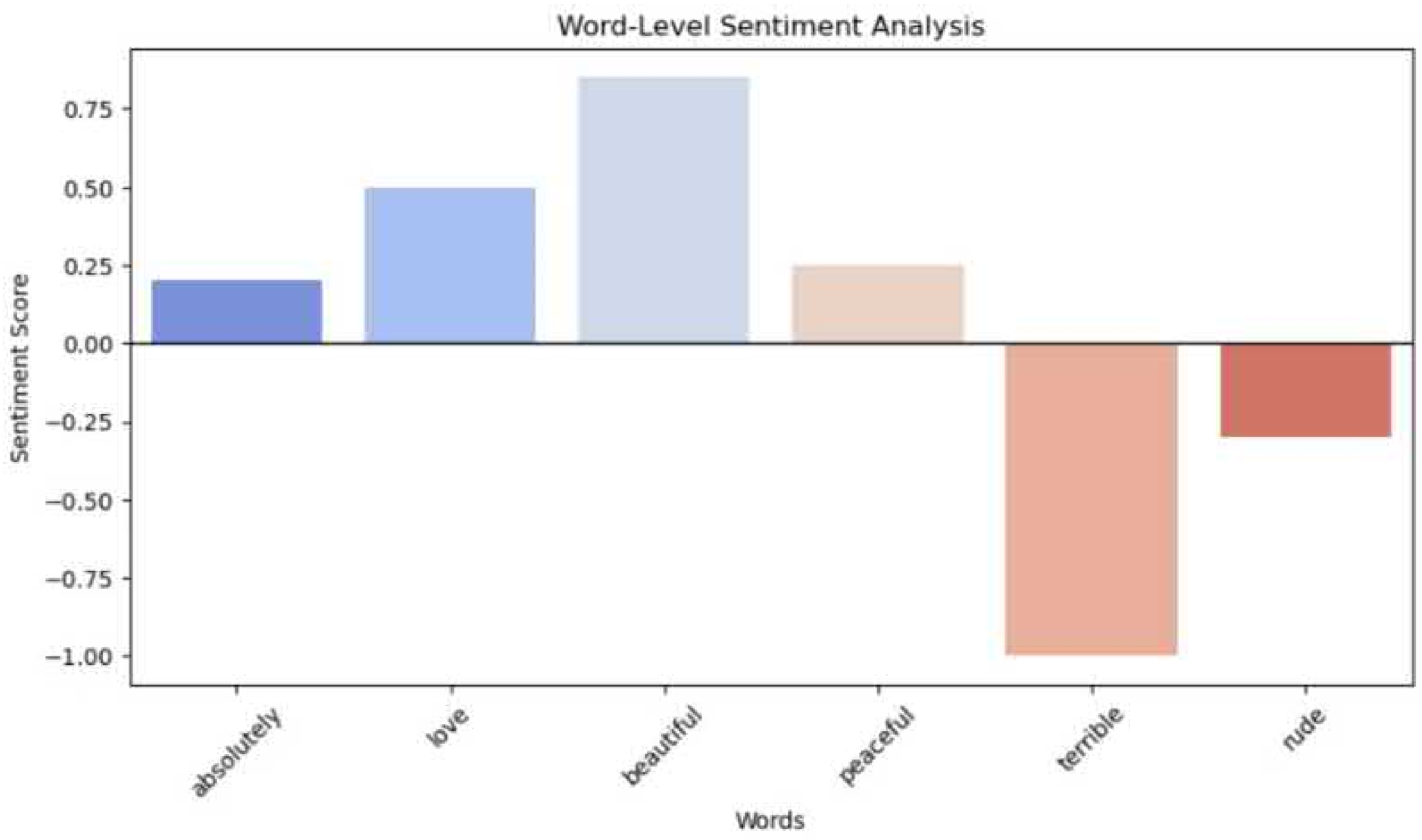


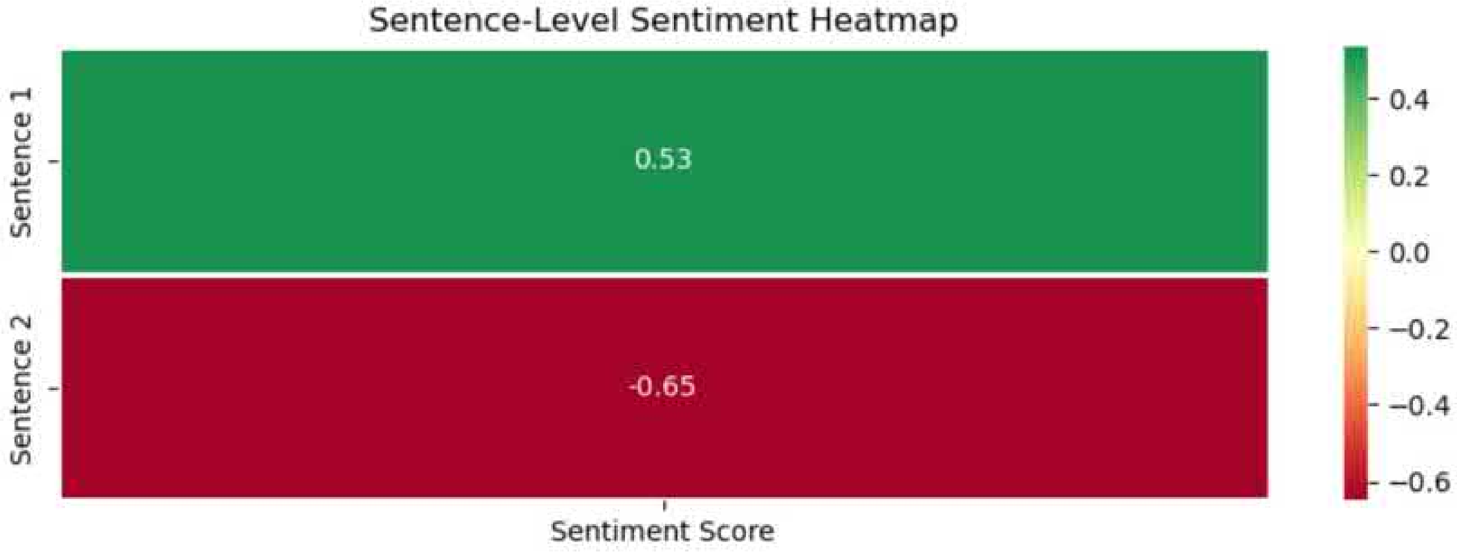
```
◆ Word-Level Sentiment Scores:
absolutely   | Sentiment Score: 0.20
love         | Sentiment Score: 0.50
beautiful    | Sentiment Score: 0.85
peaceful     | Sentiment Score: 0.25
terrible     | Sentiment Score: -1.00
rude         | Sentiment Score: -0.30

◆ Sentence-Level Sentiment Scores:
Sentence: I absolutely love the beautiful scenery and the peaceful atmosphere.

Sentiment Score: 0.53 (Positive)

Sentence: But the service was terrible, and the staff were rude and unhelpful.
Sentiment Score: -0.65 (Negative)
```

◆ **Key Takeaways:** Classify the sentiment expressed in the text as positive, negative, or neutral.

Sentiment Analysis (1)

Visualisation of Sentiments



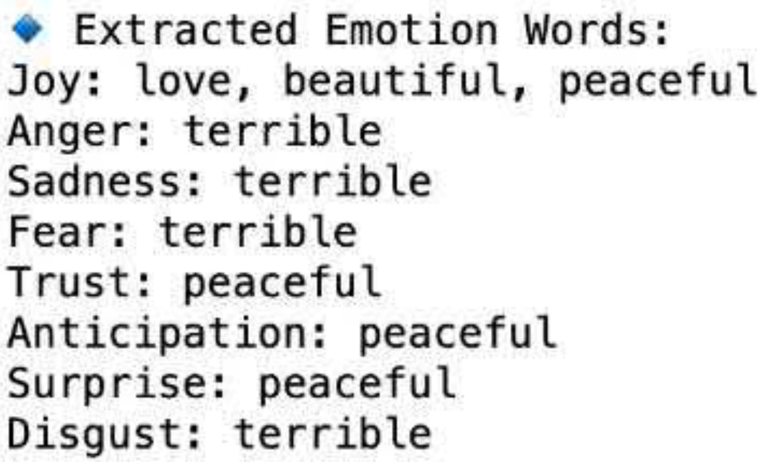


Sentiment Analysis (2)

Text: *"I absolutely love the beautiful scenery and the peaceful atmosphere. But the service was terrible, and the staff were rude and unhelpful."*

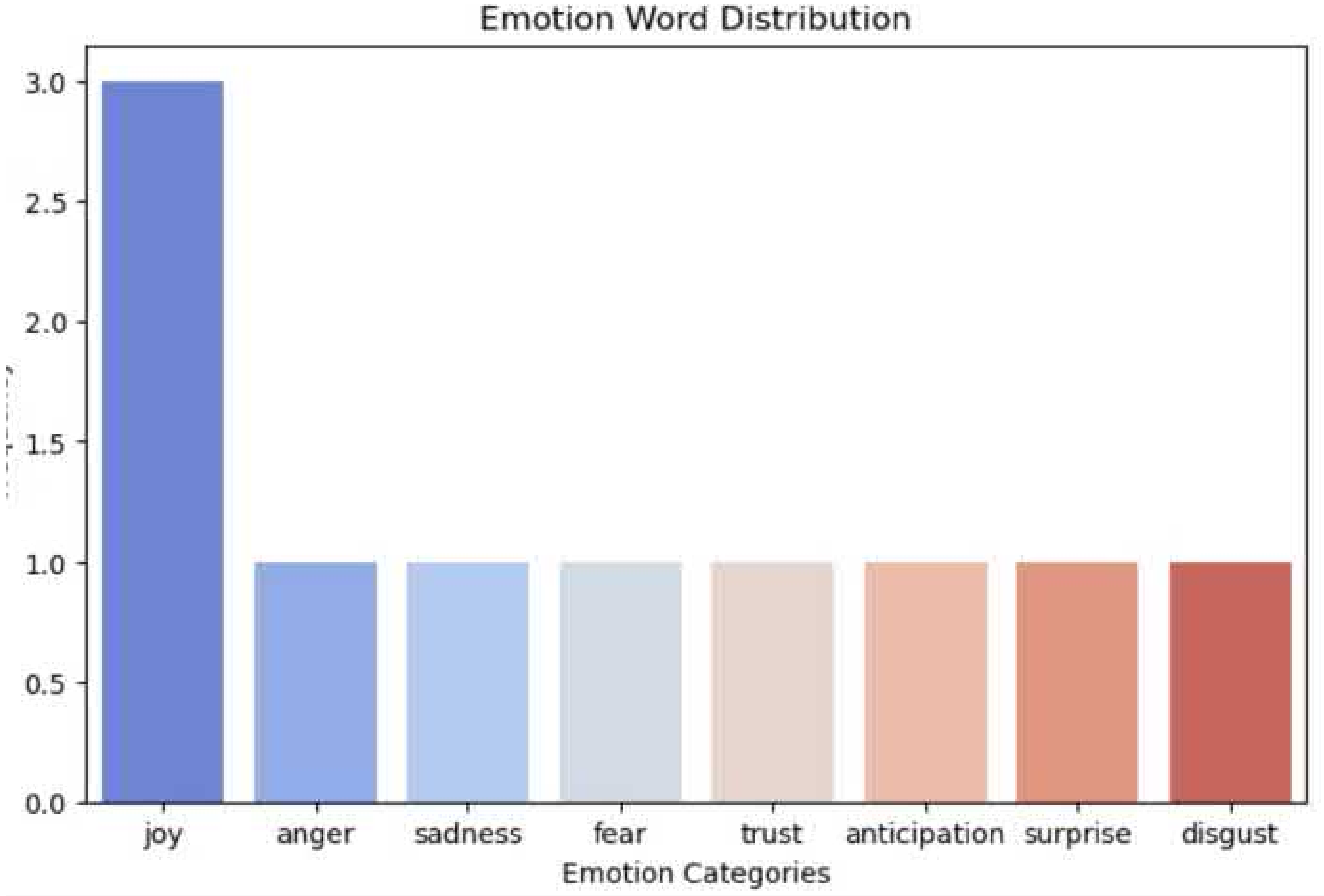
Emotion Analysis: an extension of sentiment analysis that beyond positive/negative sentiment that helps to deeper emotional context of a text rather than just its

➔ Idea: Use an emotion dictionary system: e.g., NRC (National Research Council Canada) emotion system.



```
🔹 Extracted Emotion Words:
Joy: love, beautiful, peaceful
Anger: terrible
Sadness: terrible
Fear: terrible
Trust: peaceful
Anticipation: peaceful
Surprise: peaceful
Disgust: terrible
```

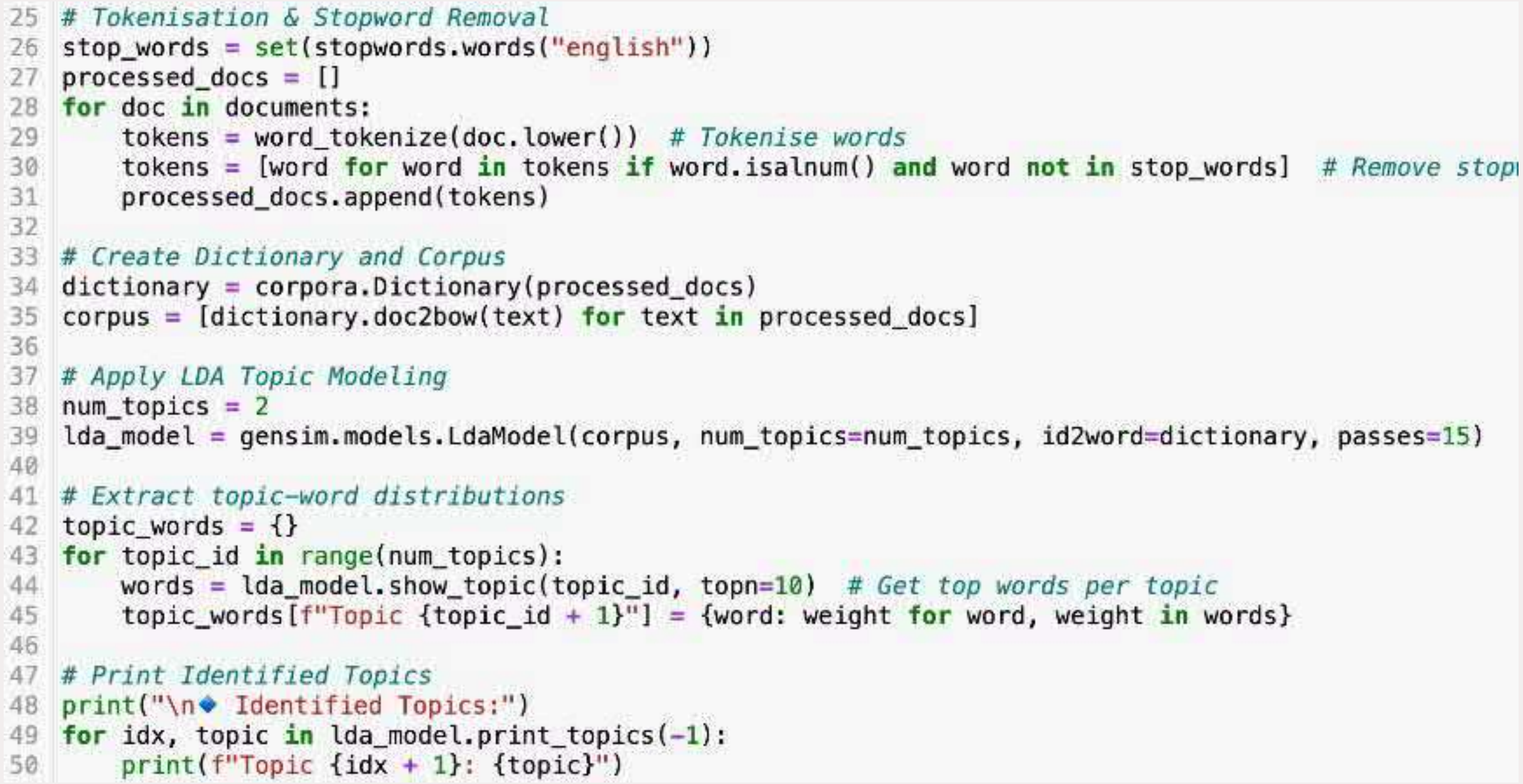
Result



Topic Modelling

Topic modelling helps discover hidden themes within a large text collection by grouping similar words into meaningful categories.

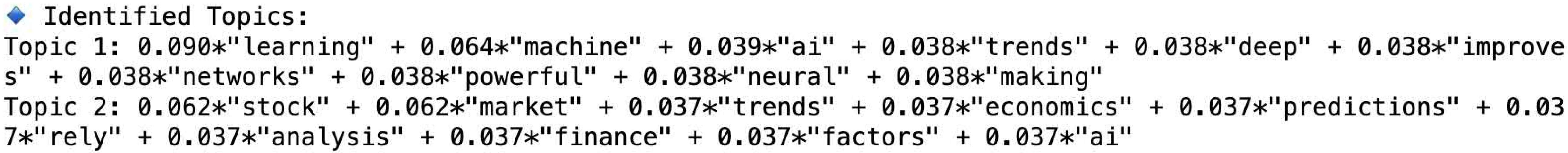
6 documents = ["Machine learning and artificial intelligence are transforming industries.", "Deep learning improves neural networks, making AI more powerful.", "Economics and finance rely on market analysis and stock predictions.", "Investors use machine learning to forecast financial trends.", "Natural Language Processing (NLP) is a key area in AI development.", "Stock market trends are influenced by political and economic factors."]



```python
# Tokenisation & Stopword Removal
stop_words = set(stopwords.words("english"))
processed_docs = []
for doc in documents:
    tokens = word_tokenize(doc.lower())  # Tokenise words
    tokens = [word for word in tokens if word.isalnum() and word not in stop_words]  # Remove stop
    processed_docs.append(tokens)

# Create Dictionary and Corpus
dictionary = corpora.Dictionary(processed_docs)
corpus = [dictionary.doc2bow(text) for text in processed_docs]

# Apply LDA Topic Modeling
num_topics = 2
lda_model = gensim.models.LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=15)

# Extract topic-word distributions
topic_words = {}
for topic_id in range(num_topics):
    words = lda_model.show_topic(topic_id, topn=10)  # Get top words per topic
    topic_words[f"Topic {topic_id + 1}"] = {word: weight for word, weight in words}

# Print Identified Topics
print("\n◆ Identified Topics:")
for idx, topic in lda_model.print_topics(-1):
    print(f"Topic {idx + 1}: {topic}")
```

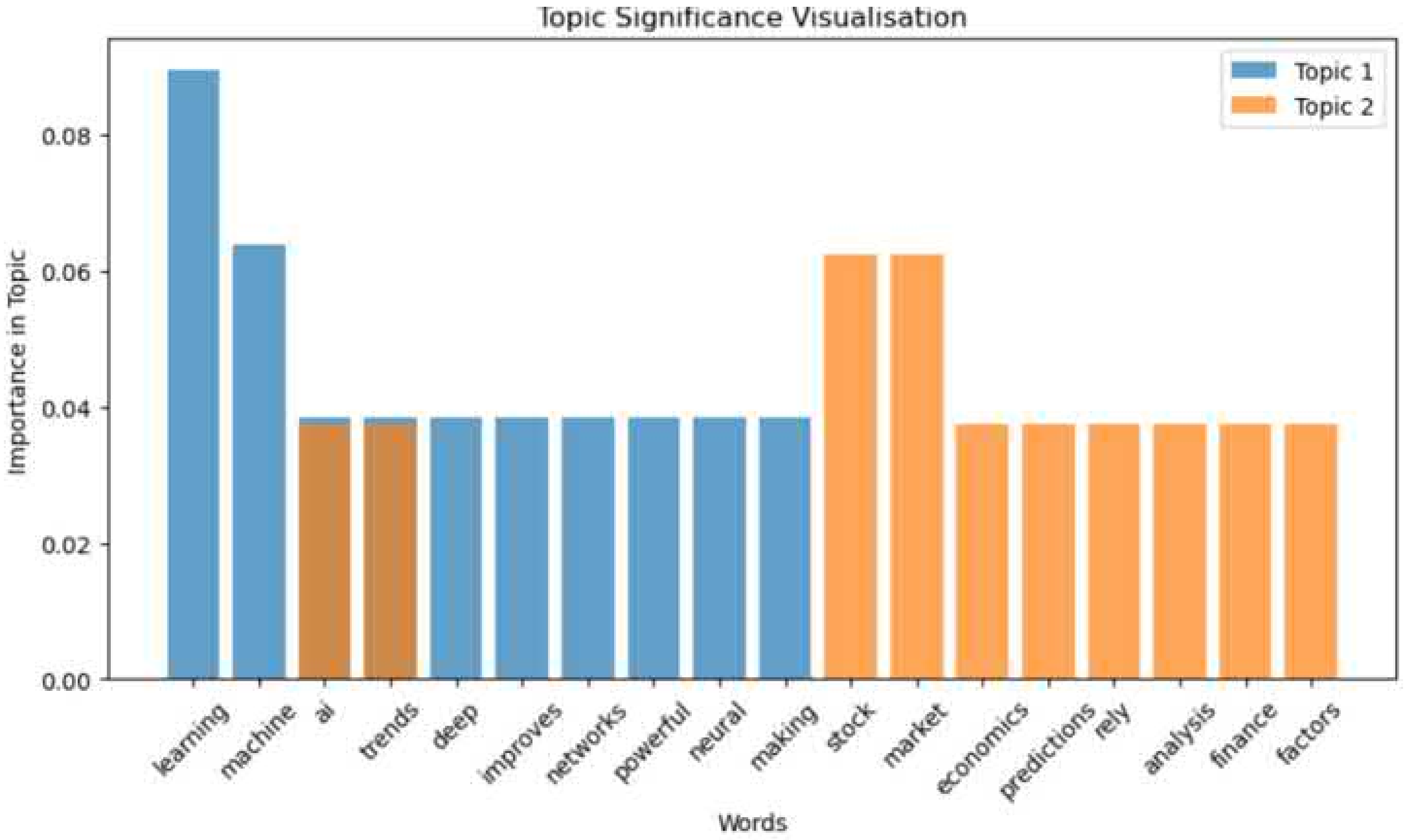
Topic Modelling

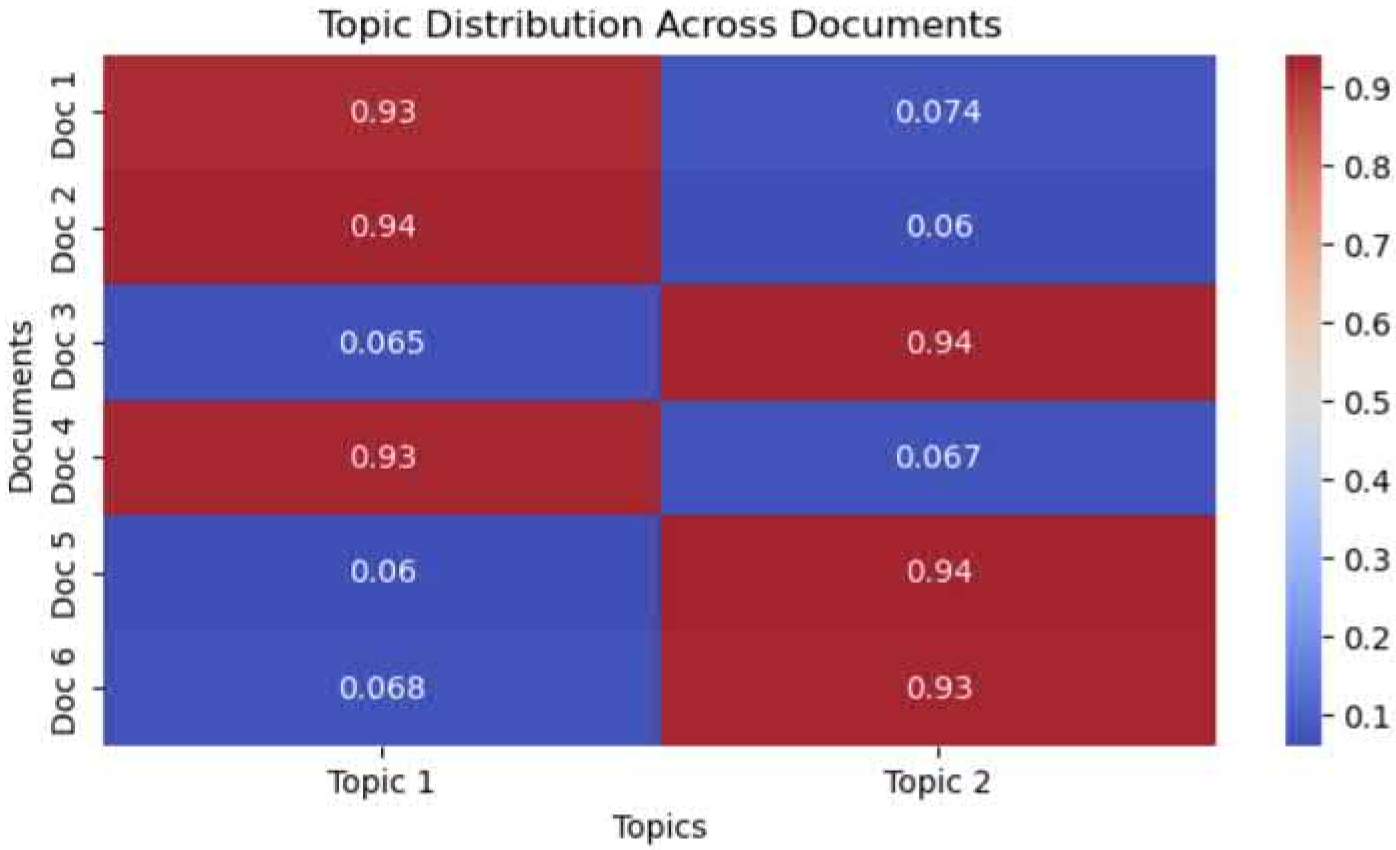
Results



```
🔹 Identified Topics:
Topic 1: 0.090*"learning" + 0.064*"machine" + 0.039*"ai" + 0.038*"trends" + 0.038*"deep" + 0.038*"improve
s" + 0.038*"networks" + 0.038*"powerful" + 0.038*"neural" + 0.038*"making"
Topic 2: 0.062*"stock" + 0.062*"market" + 0.037*"trends" + 0.037*"economics" + 0.037*"predictions" + 0.03
7*"rely" + 0.037*"analysis" + 0.037*"finance" + 0.037*"factors" + 0.037*"ai"
```

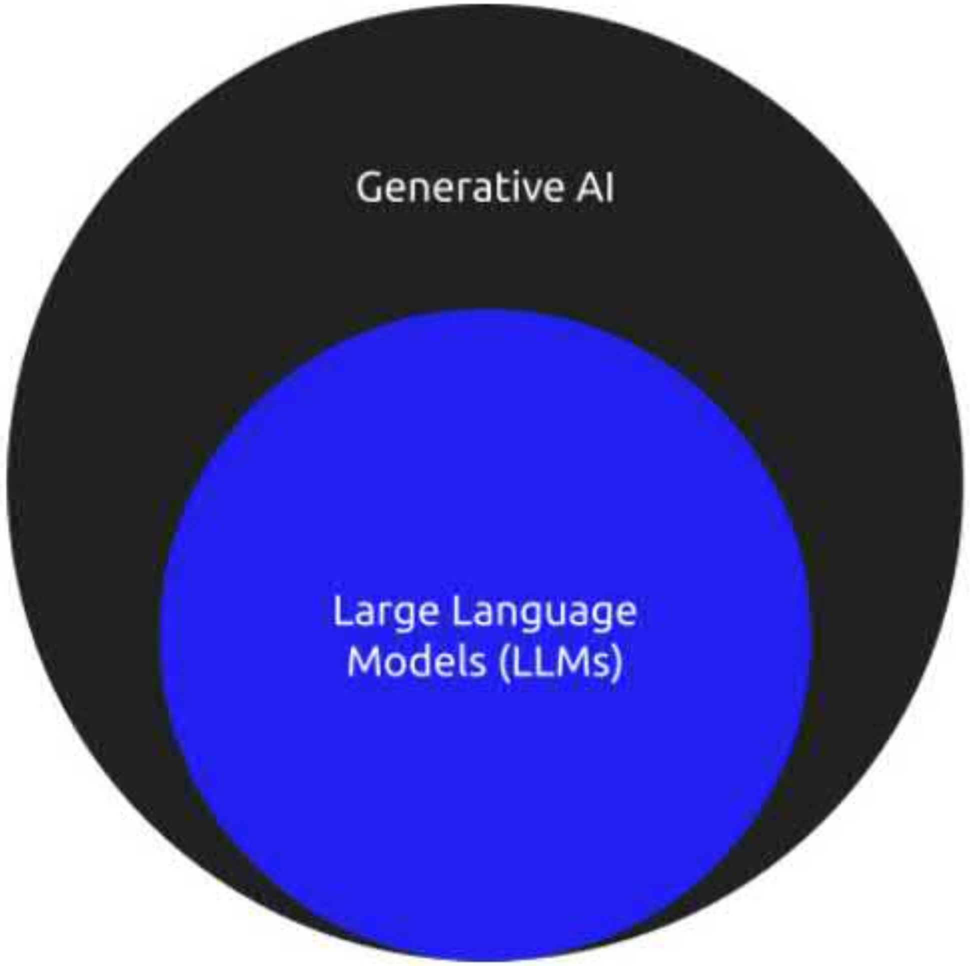
→ Topic 1: Machine Learning, Topic 2: Stock Market





Generative AI

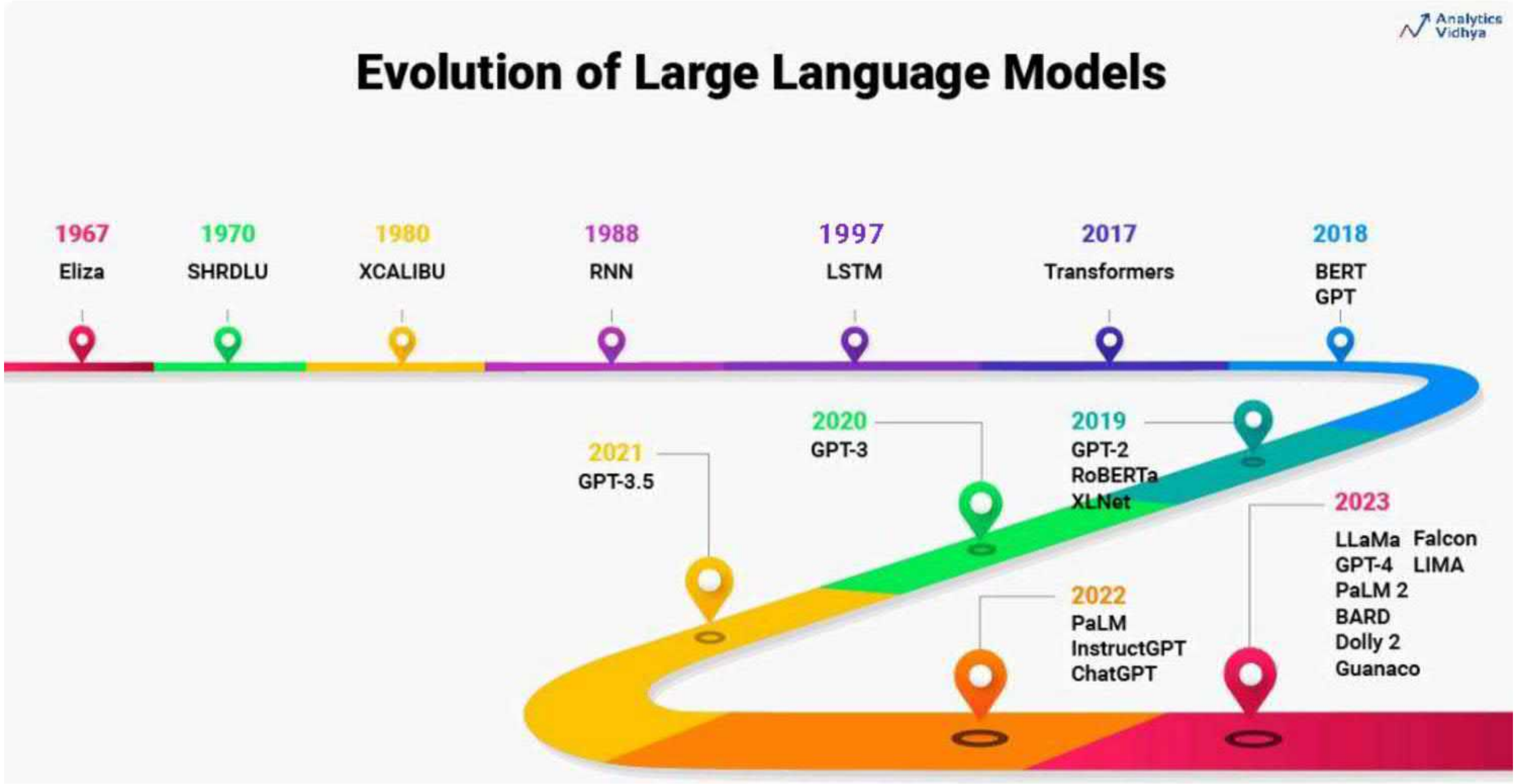
What is Large Language Model (LLM)?



- LLMs are the text-generating part of generative AI.
- LLMs are DL algorithms that can perform a variety of natural language processing (NLP) tasks.
- LLMs are trained using massive datasets — hence, **large**.

Source: On the Opportunities and Risks of Foundation Models, 2022, Arxiv

Evolution of LLMs



Source: analyticsvidhya.com (https://www.analyticsvidhya.com/blog/2023/07/beginners-guide-to-build-large-language-models-from-scratch/)

Use Cases of Generative AI (LLMs)

Text Generation

- Brainstorm new idea creation
- Assisting research strategies
- Post blogs on social media
- Translation
- Job description for hiring

Text Analysis

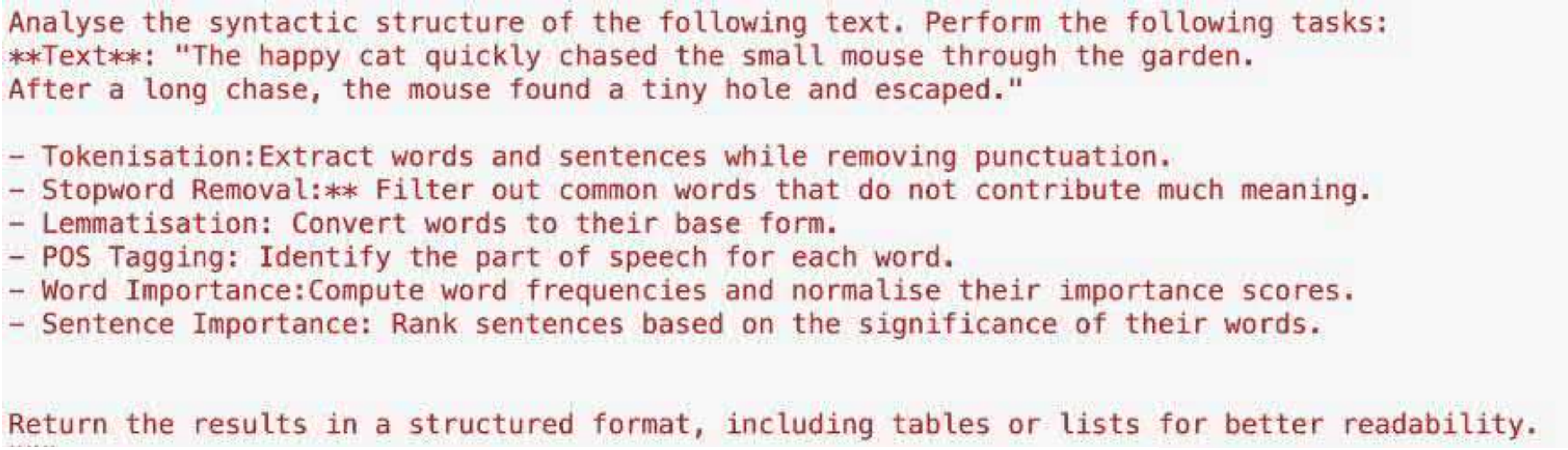
- Proofreading
- Text summarisation
- Question generation
- Skill assessment
- Customer feedback analysis
- Job matching
- Reporting

ChatBot

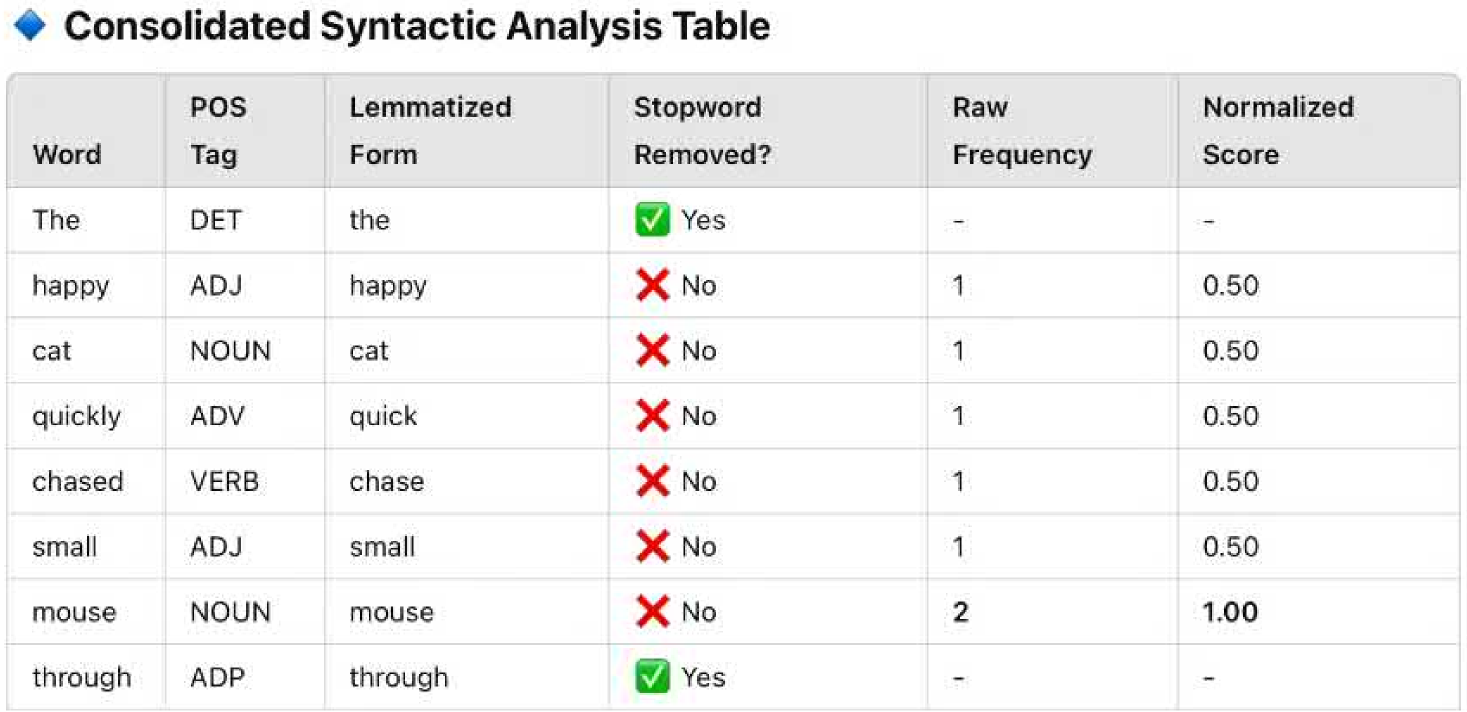
- Customer service assistance

Generative AI: Syntactic Analysis

Example: Prompt for Syntactic Structure Analysis Using ChatGPT



```
Analyse the syntactic structure of the following text. Perform the following tasks:
**Text**: "The happy cat quickly chased the small mouse through the garden.
After a long chase, the mouse found a tiny hole and escaped."

- Tokenisation:Extract words and sentences while removing punctuation.
- Stopword Removal:** Filter out common words that do not contribute much meaning.
- Lemmatisation: Convert words to their base form.
- POS Tagging: Identify the part of speech for each word.
- Word Importance:Compute word frequencies and normalise their importance scores.
- Sentence Importance: Rank sentences based on the significance of their words.


Return the results in a structured format, including tables or lists for better readability.
```

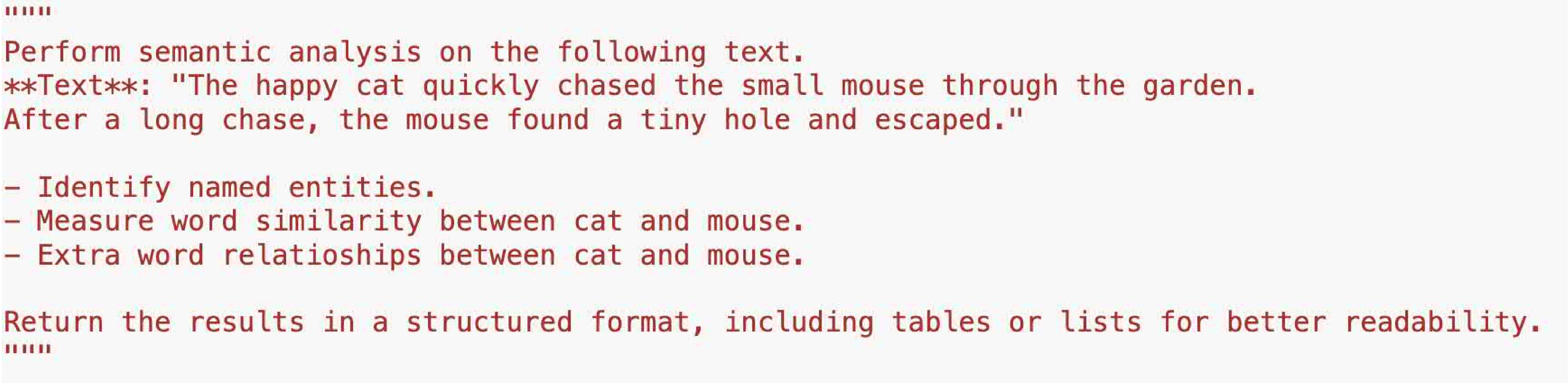


🔷 Consolidated Syntactic Analysis Table

Word	POS Tag	Lemmatized Form	Stopword Removed?	Raw Frequency	Normalized Score
The	DET	the	✅ Yes	-	-
happy	ADJ	happy	❌ No	1	0.50
cat	NOUN	cat	❌ No	1	0.50
quickly	ADV	quick	❌ No	1	0.50
chased	VERB	chase	❌ No	1	0.50
small	ADJ	small	❌ No	1	0.50
mouse	NOUN	mouse	❌ No	**2**	**1.00**
through	ADP	through	✅ Yes	-	-

Generative AI: Semantic Analysis

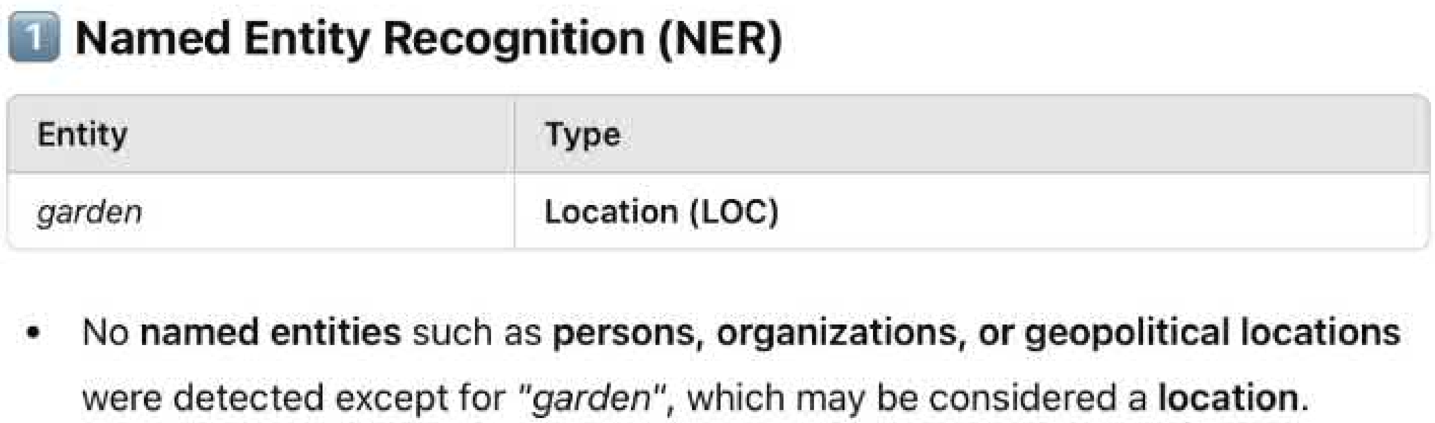
Example: Prompt for Semantic Analysis Using ChatGPT, including NET Extraction, Word Similarity, and Relationship Extraction



```
"""
Perform semantic analysis on the following text.
**Text**: "The happy cat quickly chased the small mouse through the garden.
After a long chase, the mouse found a tiny hole and escaped."

- Identify named entities.
- Measure word similarity between cat and mouse.
- Extra word relatioships between cat and mouse.

Return the results in a structured format, including tables or lists for better readability.
"""
```

Generative AI: Semantic Analysis (Output)

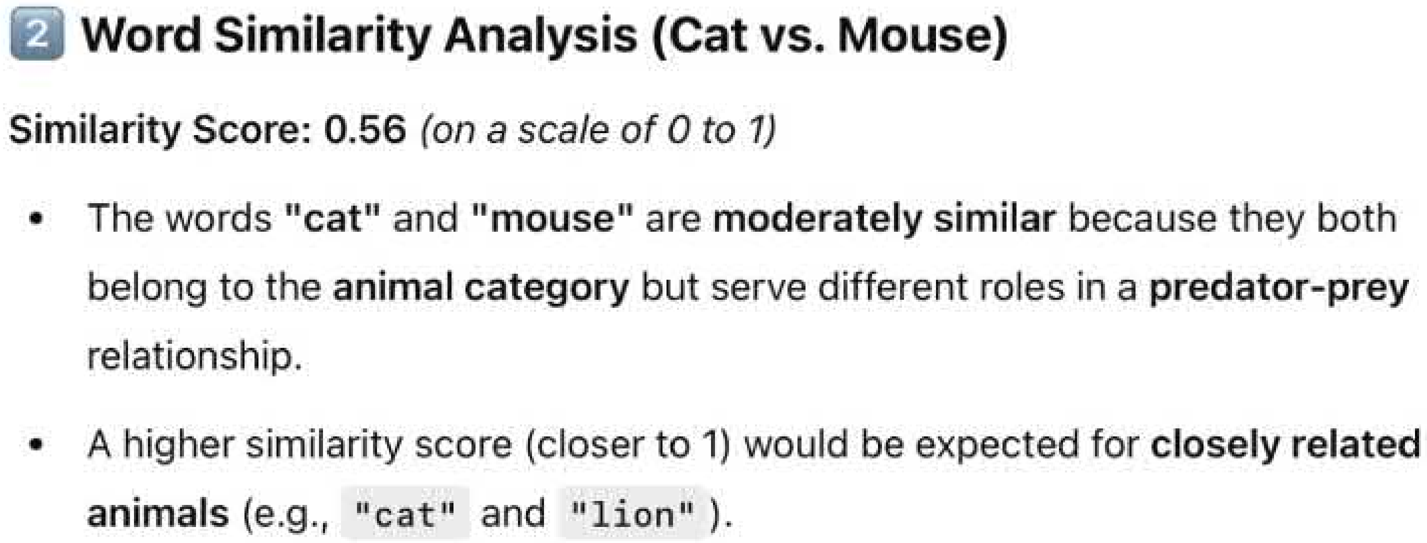
Example: Output of NER Extraction, Word Similarity, and Relationship Extraction



1️⃣ Named Entity Recognition (NER)

Entity	Type
garden	**Location (LOC)**

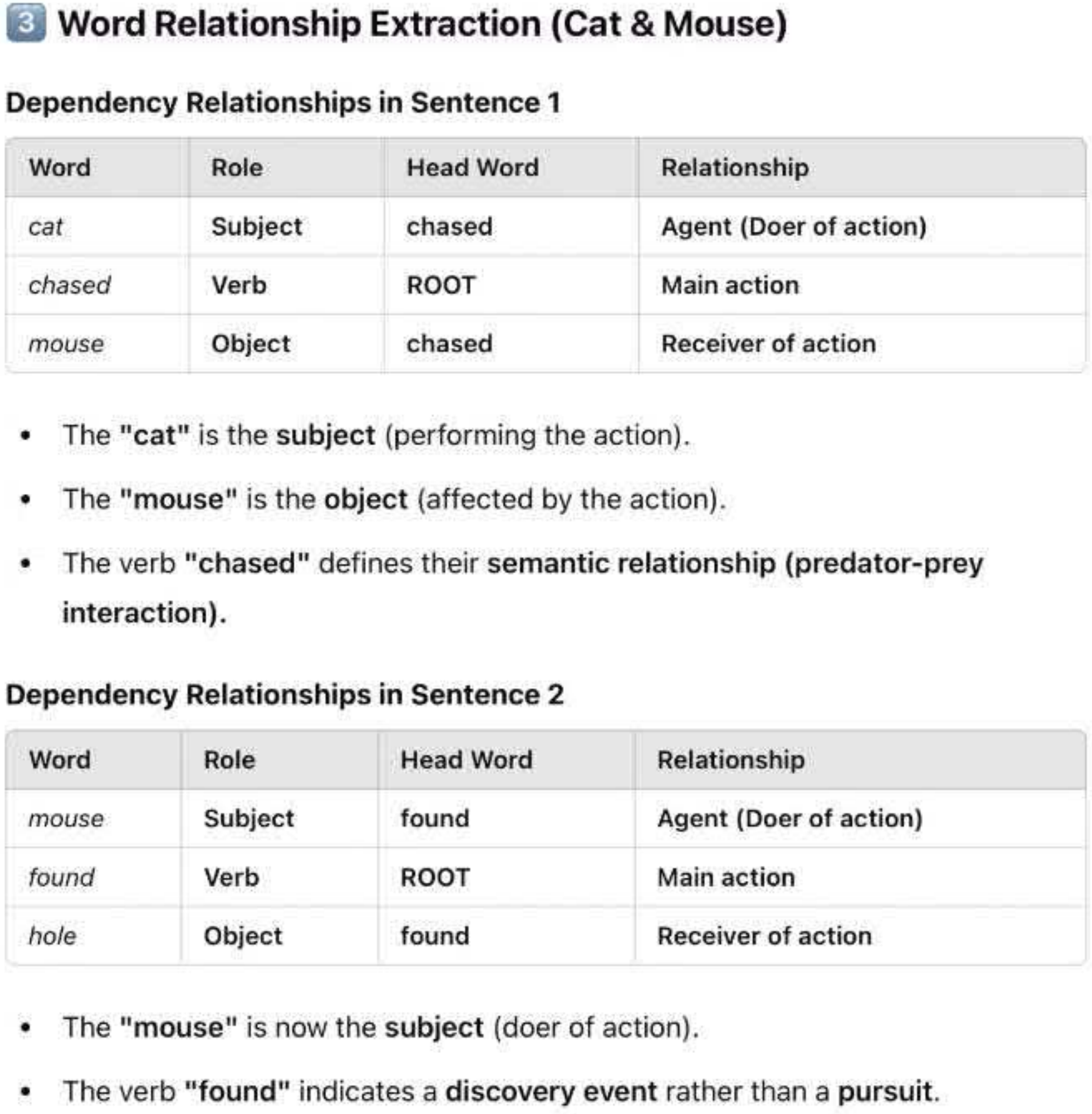
- No **named entities** such as **persons, organizations, or geopolitical locations** were detected except for *"garden"*, which may be considered a **location**.



2️⃣ Word Similarity Analysis (Cat vs. Mouse)

Similarity Score: 0.56 *(on a scale of 0 to 1)*

- The words **"cat"** and **"mouse"** are **moderately similar** because they both belong to the **animal category** but serve different roles in a **predator-prey** relationship.
- A higher similarity score (closer to 1) would be expected for **closely related animals** (e.g., `"cat"` and `"lion"`).



3️⃣ Word Relationship Extraction (Cat & Mouse)

Dependency Relationships in Sentence 1

Word	Role	Head Word	Relationship
cat	Subject	chased	Agent (Doer of action)
chased	Verb	ROOT	Main action
mouse	Object	chased	Receiver of action

- The **"cat"** is the **subject** (performing the action).
- The **"mouse"** is the **object** (affected by the action).
- The verb **"chased"** defines their **semantic relationship (predator-prey interaction).**

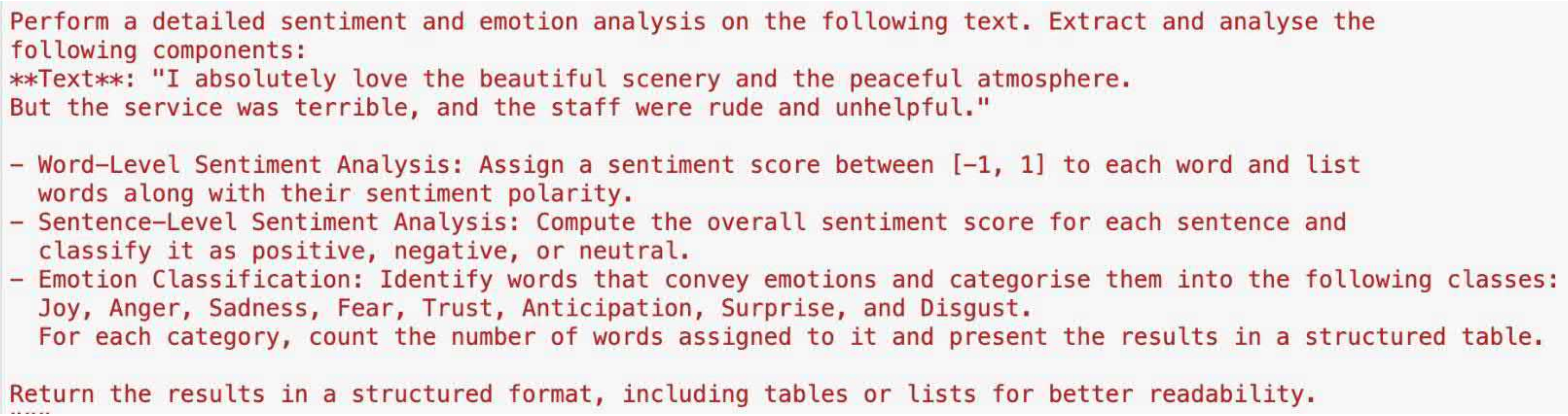
Dependency Relationships in Sentence 2

Word	Role	Head Word	Relationship
mouse	Subject	found	Agent (Doer of action)
found	Verb	ROOT	Main action
hole	Object	found	Receiver of action

- The **"mouse"** is now the **subject** (doer of action).
- The verb **"found"** indicates a **discovery event** rather than a **pursuit**.

Generative AI: Sentiment Analysis

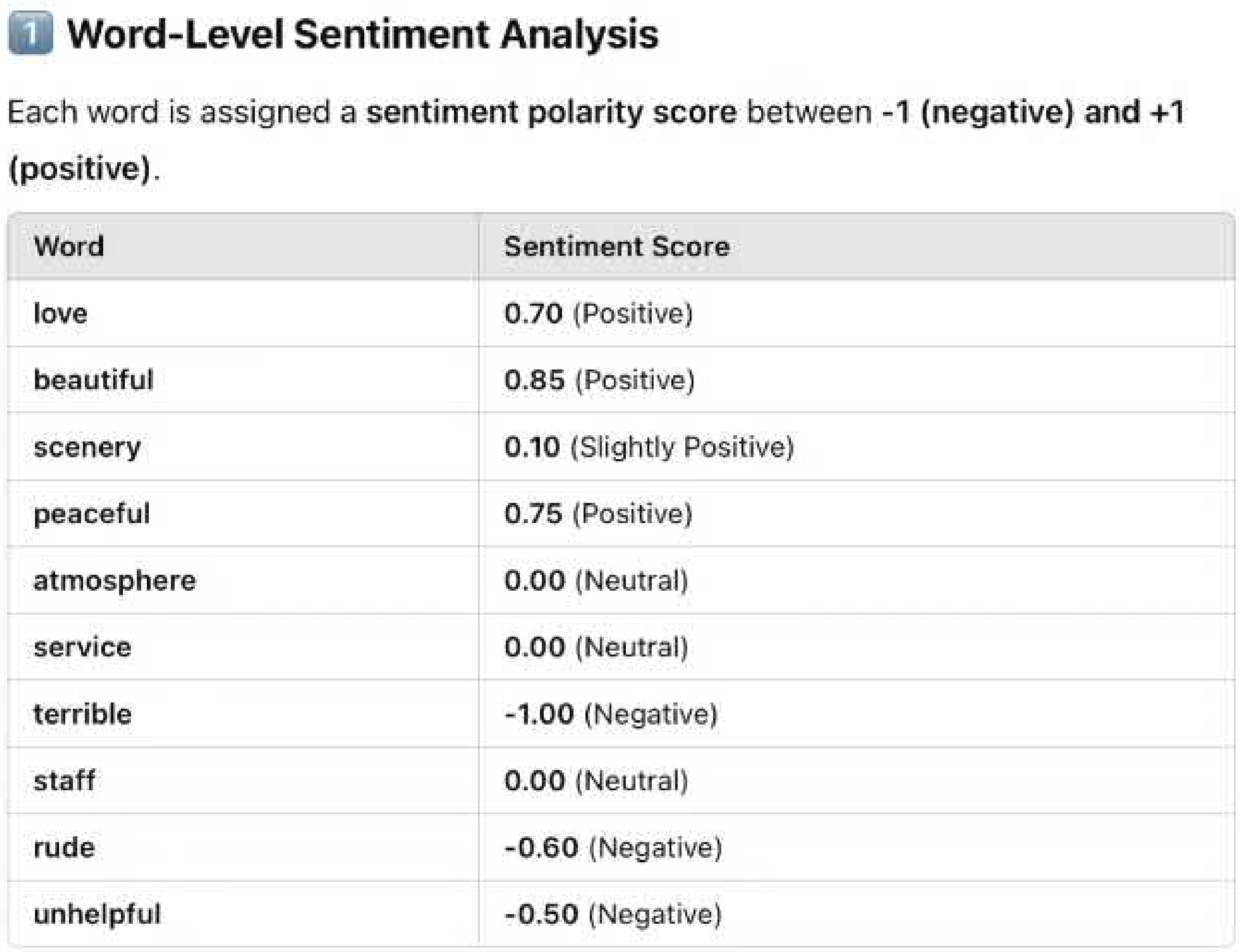
Example: Prompt for Sentiment Analysis Using ChatGPT



```
Perform a detailed sentiment and emotion analysis on the following text. Extract and analyse the
following components:
**Text**: "I absolutely love the beautiful scenery and the peaceful atmosphere.
But the service was terrible, and the staff were rude and unhelpful."

- Word-Level Sentiment Analysis: Assign a sentiment score between [-1, 1] to each word and list
  words along with their sentiment polarity.
- Sentence-Level Sentiment Analysis: Compute the overall sentiment score for each sentence and
  classify it as positive, negative, or neutral.
- Emotion Classification: Identify words that convey emotions and categorise them into the following classes:
  Joy, Anger, Sadness, Fear, Trust, Anticipation, Surprise, and Disgust.
  For each category, count the number of words assigned to it and present the results in a structured table.

Return the results in a structured format, including tables or lists for better readability.
```

Generative AI: Sentiment Analysis

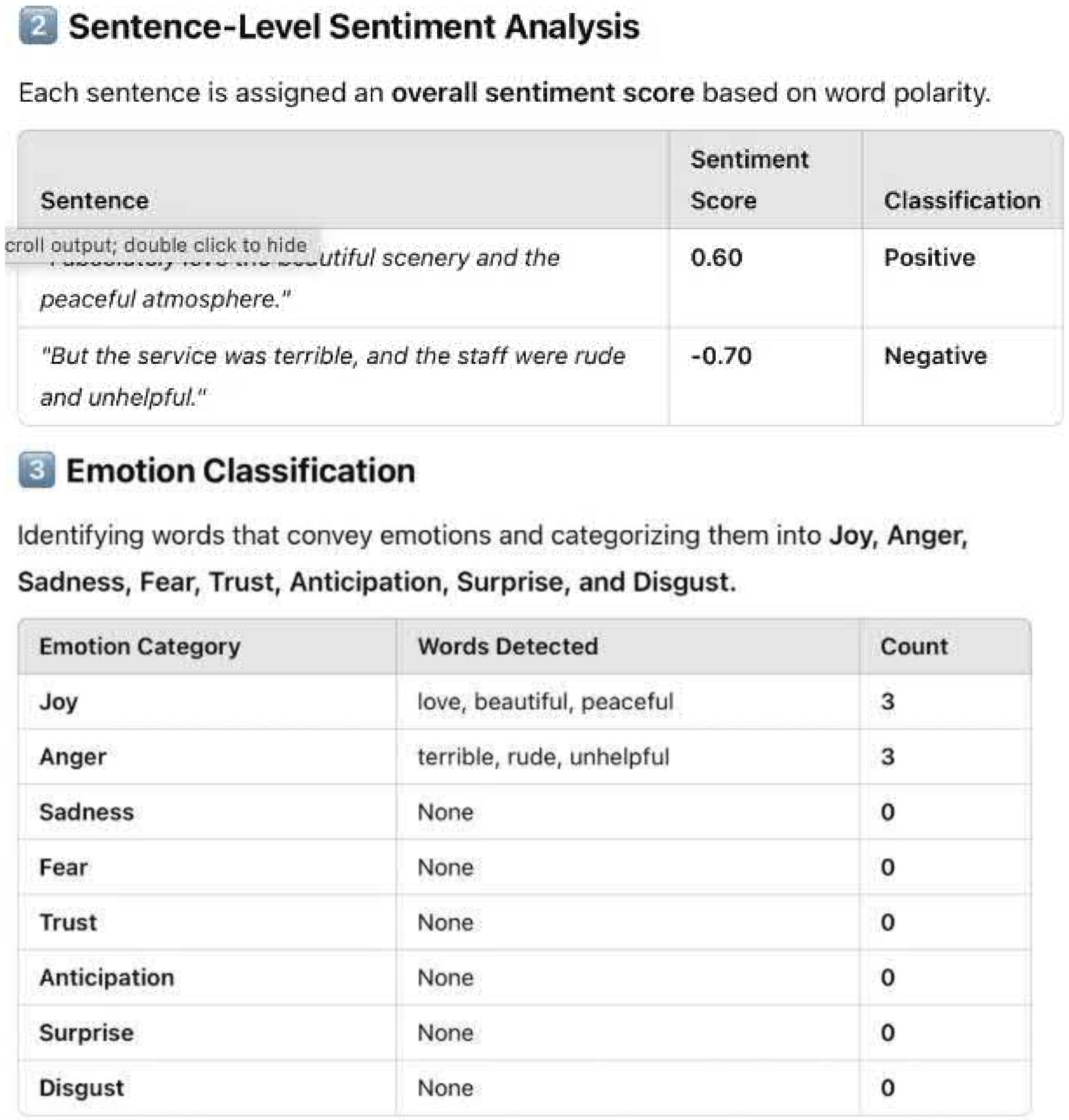
Example: Output of Sentiment Analysis



1 Word-Level Sentiment Analysis

Each word is assigned a **sentiment polarity score** between **-1 (negative) and +1 (positive)**.

Word	Sentiment Score
love	0.70 (Positive)
beautiful	0.85 (Positive)
scenery	0.10 (Slightly Positive)
peaceful	0.75 (Positive)
atmosphere	0.00 (Neutral)
service	0.00 (Neutral)
terrible	-1.00 (Negative)
staff	0.00 (Neutral)
rude	-0.60 (Negative)
unhelpful	-0.50 (Negative)



2 Sentence-Level Sentiment Analysis

Each sentence is assigned an **overall sentiment score** based on word polarity.

Sentence	Sentiment Score	Classification
croll output; double click to hide *"I absolutely love the beautiful scenery and the peaceful atmosphere."*	0.60	Positive
"But the service was terrible, and the staff were rude and unhelpful."	-0.70	Negative

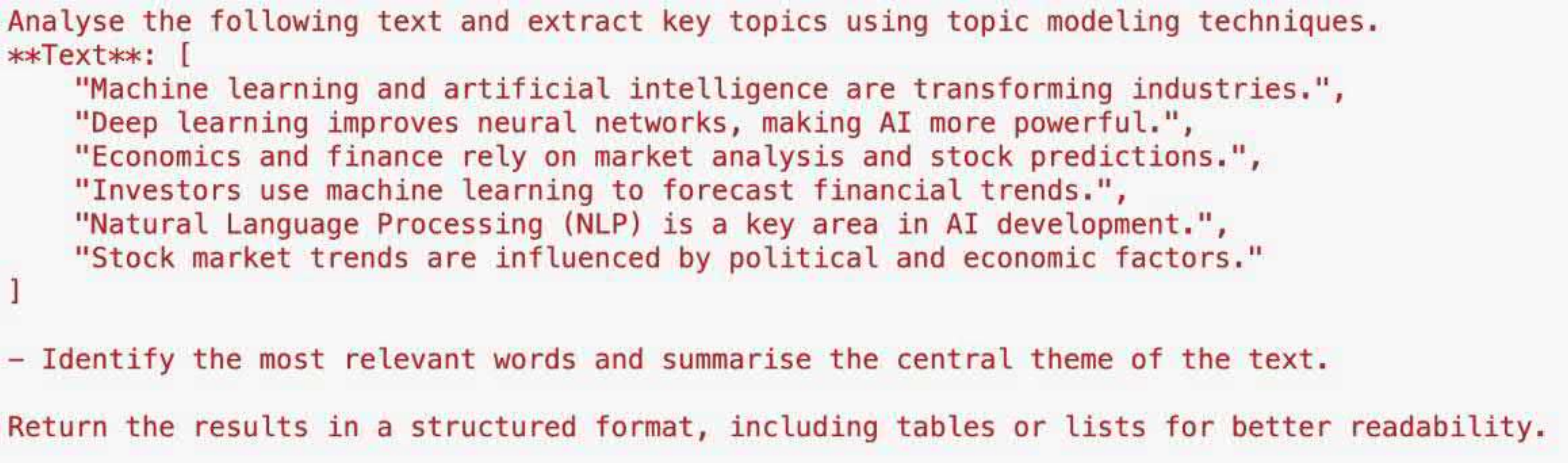
3 Emotion Classification

Identifying words that convey emotions and categorizing them into **Joy, Anger, Sadness, Fear, Trust, Anticipation, Surprise, and Disgust.**

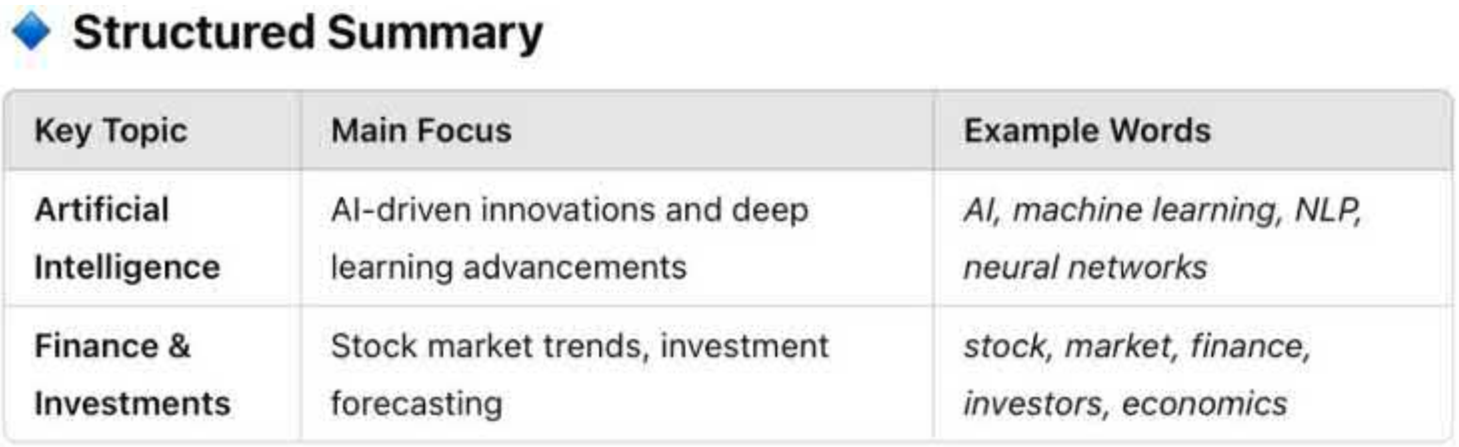
Emotion Category	Words Detected	Count
Joy	love, beautiful, peaceful	3
Anger	terrible, rude, unhelpful	3
Sadness	None	0
Fear	None	0
Trust	None	0
Anticipation	None	0
Surprise	None	0
Disgust	None	0

Generative AI: Topic Modelling

Example: Prompt for Topic Modelling Using ChatGPT



```
Analyse the following text and extract key topics using topic modeling techniques.
**Text**: [
    "Machine learning and artificial intelligence are transforming industries.",
    "Deep learning improves neural networks, making AI more powerful.",
    "Economics and finance rely on market analysis and stock predictions.",
    "Investors use machine learning to forecast financial trends.",
    "Natural Language Processing (NLP) is a key area in AI development.",
    "Stock market trends are influenced by political and economic factors."
]

- Identify the most relevant words and summarise the central theme of the text.

Return the results in a structured format, including tables or lists for better readability.
```



🔷 Structured Summary

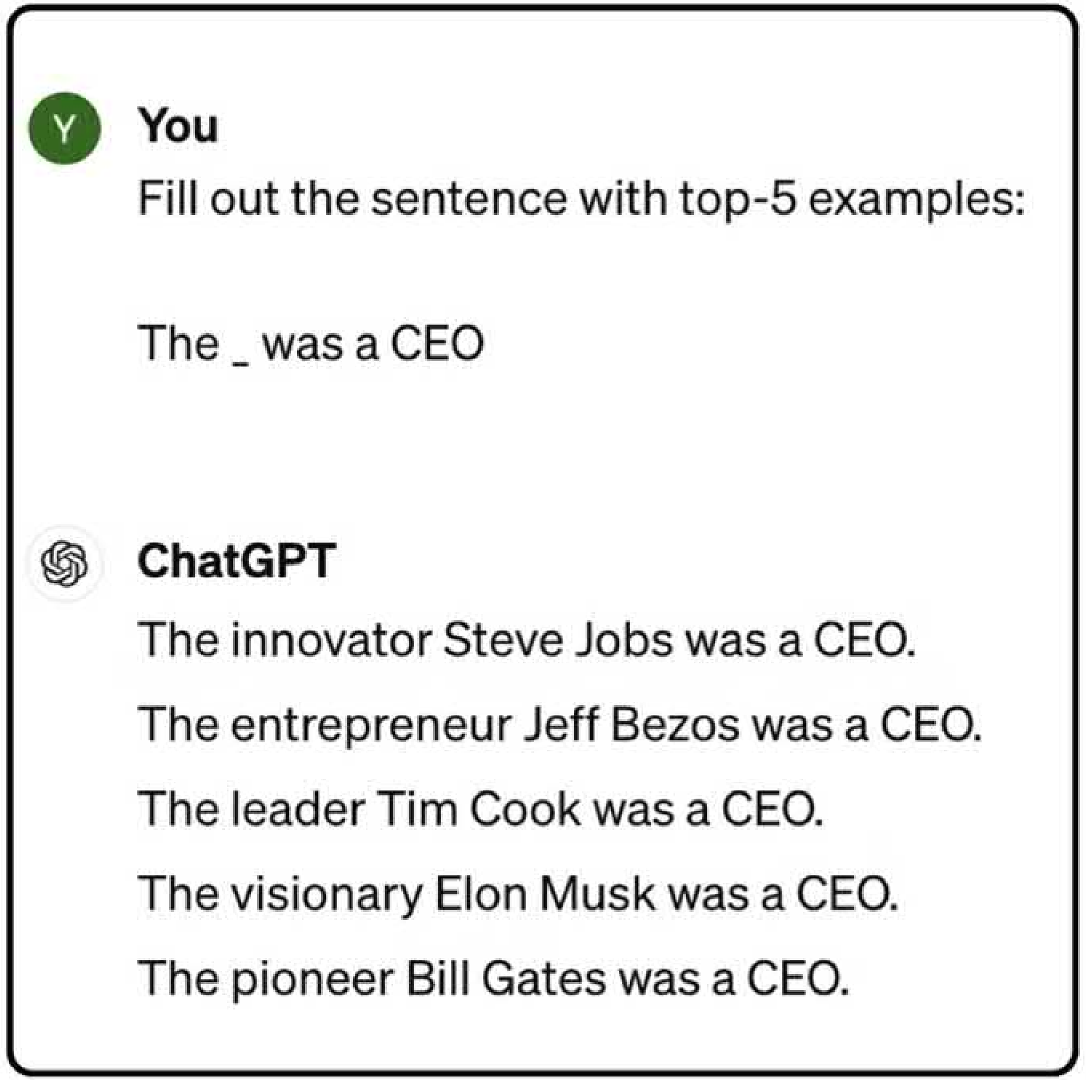
Key Topic	Main Focus	Example Words
Artificial Intelligence	AI-driven innovations and deep learning advancements	*AI, machine learning, NLP, neural networks*
Finance & Investments	Stock market trends, investment forecasting	*stock, market, finance, investors, economics*

Traditional Approaches vs. LLMs

	Traditional NLP	**LLMs (Generative AI)**
Strengths	- Well-defined pipelines - Easier to explain - Fast for simple tasks - Customisable for specific domains	- Deep contextual understanding - Multi-tasking - Less manual setup - State-of-the-art performance
Weaknesses	- Limited flexibility - Task-specific - Requires more manual effort for complex tasks	- Resource-intensive - Less transparent decision-making - Risk of overfitting or bias
Computational Resources	Lower resource usage, faster setup for smaller tasks	High resource usage (GPU/TPU), slower inference for larger models
Interpretability	High - Transparent decision-making (e.g., for rule-based tasks)	Low - Often seen as a 'black box' with difficult-to-understand reasoning
Customisability	High – Can tailor for specific tasks and domains	Lower – Pretrained models are harder to fine-tune for specific domains
Performance	Good for straightforward tasks, may struggle with complexity	Excellent for complex, nuanced, and ambiguous text analysis
Explainability	High – Easier to explain results and processes	Low – Outputs are often difficult to interpret or explain

Conclusion

Generative AI: Concerns



You

Fill out the sentence with top-5 examples:

The _ was a CEO

ChatGPT

The innovator Steve Jobs was a CEO.

The entrepreneur Jeff Bezos was a CEO.

The leader Tim Cook was a CEO.

The visionary Elon Musk was a CEO.

The pioneer Bill Gates was a CEO.

What do we see?

However, LLMs are improving with bias reduction via refinement and methodologies like Reinforcement Learning from Human Feedback (RLHF).

Conclusion

NLP Fundamentals

NLP has evolved from rule-based syntactic analysis to sophisticated semantic understanding, enabling transformative applications across industries and disciplines.

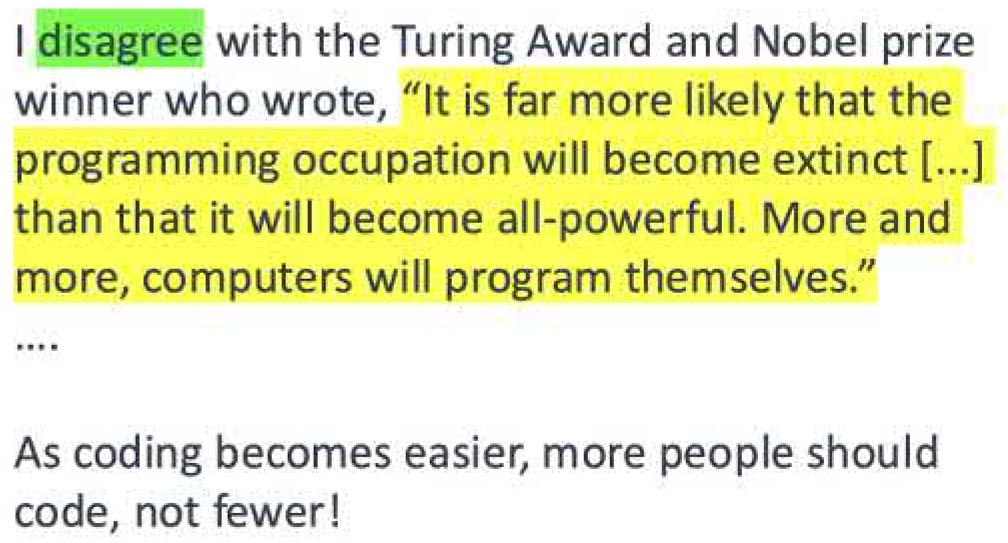
Generative AI Revolution

Generative AI models have fundamentally transformed NLP capabilities, capabilities, delivering unprecedented accuracy, fluency, and contextual contextual awareness in language processing tasks.

Future Applications

The integration of NLP and generative AI will continue to revolutionise text revolutionise text analysis, creating powerful opportunities for human-like human-like language understanding and generation.

Andrew Ng on March 12, 2025
(https://www.deeplearning.ai/the-batch/issue-292/ :



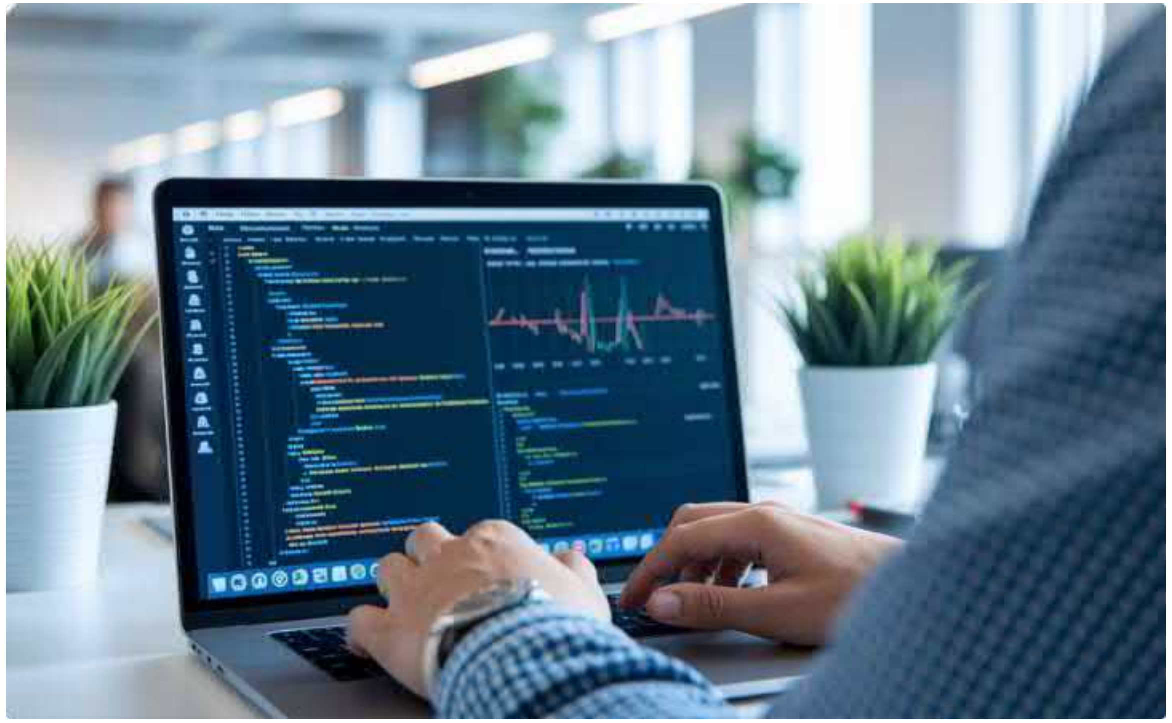
I disagree with the Turing Award and Nobel prize winner who wrote, “It is far more likely that the programming occupation will become extinct [...] than that it will become all-powerful. More and more, computers will program themselves.”

....

As coding becomes easier, more people should code, not fewer!

Thank You

> **AI is not just transforming how we analyse text—it's fundamentally changing how humans communicate, create, and connect with each other.**



Explore Our Resources

Access our Jupyter notebook demo code:

https://github.com/Yongbinkang/nlp_demo/

Contact information: ykang@swin.edu.au