In this session, we are going to explore how we can use computer tools to make sense of biological information written in text: like research papers, medical notes, or scientific databases. This kind of text data is called "unstructured data", because it is not organized like a spreadsheet or table.

There is an explosion of biological information published daily: thousands of new studies, reports, and articles. It is way too much for any human to read or analyze alone. So we use Natural Language Processing (NLP), a field of artificial intelligence, to help computers read and understand human language.

Retrieving real scientific articles from PubMed (a giant online library of life science research). Searching for papers about CRISPR gene editing, a powerful tool that can edit genes in living organisms.

We will use NLP techniques to pull out key topics, terms, and patterns from the text. We will use visualization tools to create simple charts or graphs that show us what is trending, what terms often appear together, or how topics have changed over time.

Why it is necessary? How modern tools can help summarize huge amounts of scientific information. How you can use text data to discover insights, trends, and connections in biology and medicine.

```
#Libraries
import pandas as pd
import numpy as np

# Bioinformatics tools
from Bio import Entrez, SeqIO  # Correct Biopython imports
from Bio.Medline import read as read_medline  # For Medline

# NLP and text processing
import spacy
from spacy import displacy
from wordcloud import WordCloud

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Machine learning
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.feature_extraction.text import TfidfVectorizer
```

This practical session will guide you through qualitative analysis of bioinformatics data using NLP and visualization techniques. We will work with PubMed abstracts and biological text data to extract meaningful insights.

```
# 1. SETUP VISUALIZATION ENVIRONMENT
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

# Set modern plotting style (correct for Seaborn 0.12+)
sns.set_theme(style="whitegrid", palette="husl")  # Primary style command
plt.rcParams['figure.facecolor'] = 'white'  # Ensure white background

# Check available styles (updated method)
print("Available matplotlib styles:", plt.style.available)
print("\nAvailable seaborn style contexts:", ["darkgrid", "whitegrid", "dark", "white", "ticks"])
print("Available seaborn palettes:", list(sns.palettes.SEABORN_PALETTES.keys()))

# 2. LOAD NLP MODEL
import spacy
try:
    nlp = spacy.load("en_core_web_sm")
except OSError:
    print("Downloading language model...")
    !pip install -U spacy
    !python -m spacy download en_core_web_sm
    nlp = spacy.load("en_core_web_sm")

# 3. SET NCBI EMAIL (REQUIRED)
from Bio import Entrez
Entrez.email = "nirajkc00@gmail.com"  # Replace with your real email

print("\n✓ Setup complete! Ready for bioinformatics analysis.")
```

```
Available matplotlib styles: ['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'da

Available seaborn style contexts: ['darkgrid', 'whitegrid', 'dark', 'white', 'ticks']
Available seaborn palettes: ['deep', 'deep6', 'muted', 'muted6', 'pastel', 'pastel6', 'bright', 'bright6', 'dark', 'dark6', 'colorbl
```

```
✓ Setup complete! Ready for bioinformatics analysis.
```

Fetching Biological Literature Data from PubMed Let's start by retrieving some recent publications about "CRISPR gene editing":

```
def fetch_pubmed_abstracts(search_term, max_results=10):
    """Fetch abstracts from PubMed"""
    handle = Entrez.esearch(db="pubmed", term=search_term, retmax=max_results)
    record = Entrez.read(handle)
    id_list = record["IdList"]

    abstracts = []
    for pubmed_id in id_list:
        handle = Entrez.efetch(db="pubmed", id=pubmed_id, rettype="medline", retmode="text")
        record = read_medline(handle)
        abstracts.append({
            "title": record.get("TI", ""),
            "abstract": record.get("AB", ""),
            "authors": record.get("AU", []),
            "journal": record.get("JT", ""),
            "pub_date": record.get("DP", "")
        })
    return pd.DataFrame(abstracts)

# Fetch sample data
crispr_df = fetch_pubmed_abstracts("CRISPR gene editing", 15)
crispr_df.head(10)
```

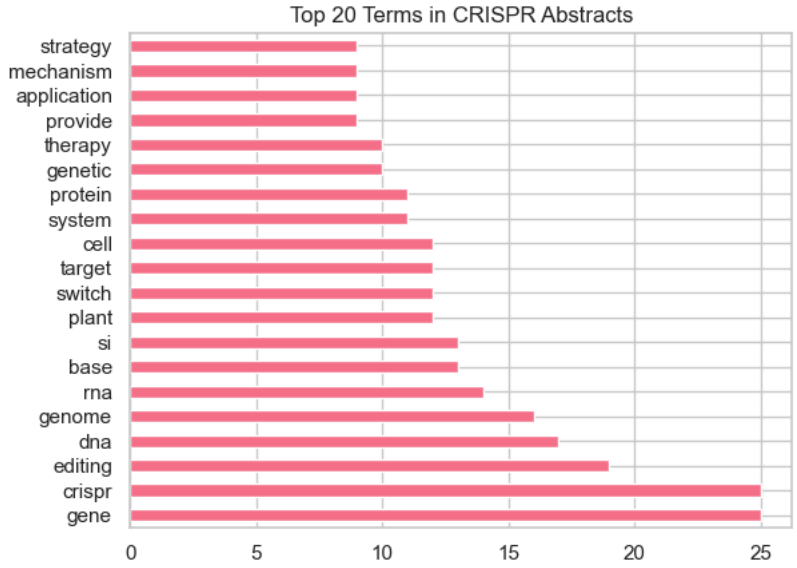
	title	abstract	authors	journal	pub_date
0	Gene therapy and genome editing for lipoprotei...	Genetic factors play a critical role in the de...	[Gurevitz C, Bajaj A, Khera AV, Do R, Schunker...	European heart journal	2025 Jul 2
1	Development of a CRISPR/Cas9 RNP-mediated gene...	A thermophilic strain of Paecilomyces variotii...	[Han HG, Nandre R, Eom H, Choi YJ, Ro HS]	Journal of microbiology (Seoul, Korea)	2025 Jun
2	SWITCHER, a CRISPR-inducible floxed wild-type ...	Although several Cre-regulated CRISPR/Cas plat...	[Schuster B, Dobiasovska I, Curcic J, Pajer P,...	Communications biology	2025 Jul 2
3	Mechanisms and engineering of a miniature type...	Type V CRISPR-Cas12 systems are highly diverse...	[Fu W, Ma J, Wang Z, Tang N, Pan D, Su M, Wu Z...	Nature communications	2025 Jul 1
4	Scalable modulation of CRISPR–Cas enzyme activ...	The regulation of CRISPR–Cas activity is criti...	[Hu M, Zhang B, Shan Y, Cao F, Wang Y, Qi W, W...	Nature communications	2025 Jul 1
5	Engineered Sdd7 cytosine base editors with enh...	Cytosine base editors (CBEs) revolutionize gen...	[Hwang HY, Lee M, Yi H, Seok C, Lim K, Na YR, ...	Nature communications	2025 Jul 1

Text Preprocessing with SpaCy

Let's clean and analyze the text:

```
This code helps the computer clean and simplify the text from scientific article summaries (called abstracts).
It uses a language tool called spaCy to go through each abstract and remove common words (like "the" or "is"), punctuation, and reduce v
Then, it combines all the cleaned-up text and counts which words appear the most.
Finally, it creates a bar chart to show the top 20 most frequently used words in CRISPR-related research.
This helps us quickly see what topics or terms are commonly discussed in these studies.
```

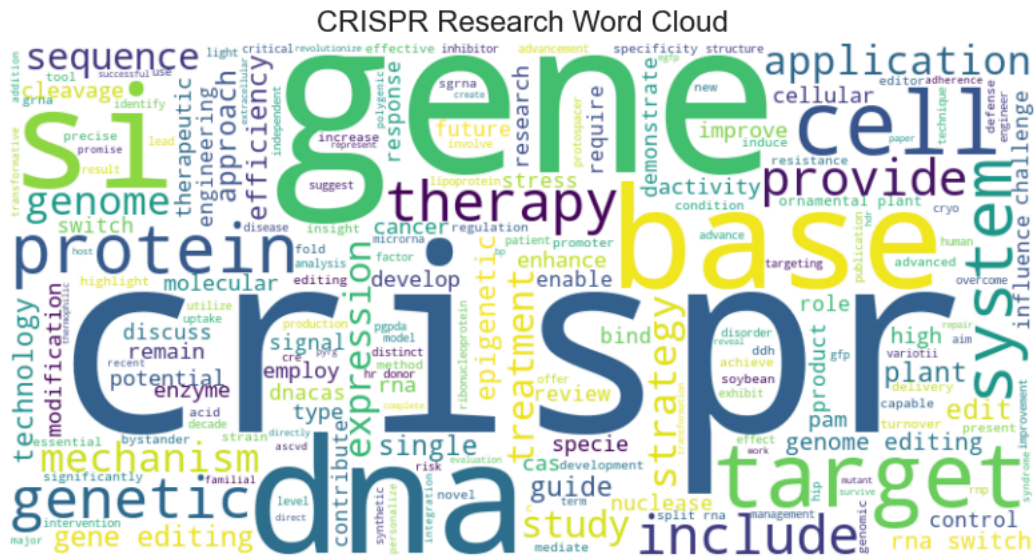
```
def process_text(text):
    """Process text with spaCy"""
    doc = nlp(text)
    # Remove stopwords, punctuation, and lemmatize
    tokens = [token.lemma_.lower() for token in doc
              if not token.is_stop and not token.is_punct and token.is_alpha]
    return " ".join(tokens)

# Apply to abstracts
crispr_df["processed_abstract"] = crispr_df["abstract"].apply(process_text)

# Show most common words
word_freq = pd.Series(" ".join(crispr_df["processed_abstract"]).split()).value_counts()[:20]
word_freq.plot(kind="barh", title="Top 20 Terms in CRISPR Abstracts");
```



Visualizing Concepts with Word Clouds

This code takes all the cleaned-up words from the CRISPR research summaries and combines them into one big block of text.
Then it uses a tool called WordCloud to create a visual image where the most frequently used words appear larger.
The result is a word cloud, which gives a quick, easy-to-understand overview of the main topics and keywords in CRISPR research.
It is like a visual summary of what is being talked about the most in those articles.

```
text = " ".join(crispr_df["processed_abstract"])
wordcloud = WordCloud(width=800, height=400, background_color="white").generate(text)

plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("CRISPR Research Word Cloud", size=16);
```



Named Entity Recognition for Biological Terms

Extract genes, proteins, and other entities

This part of the code is trying to find important biological terms in a scientific summary: like the names of genes, proteins, chemicals
It uses a language tool (spaCy) that can recognize these types of scientific entities in text.
First, it looks at the very first abstract in the dataset and pulls out any relevant biological names.
Then, it highlights these terms visually, right inside the text, so you can easily see what key scientific concepts are being discussed.
It is like having a digital highlighter that automatically marks all the important biological words for you.

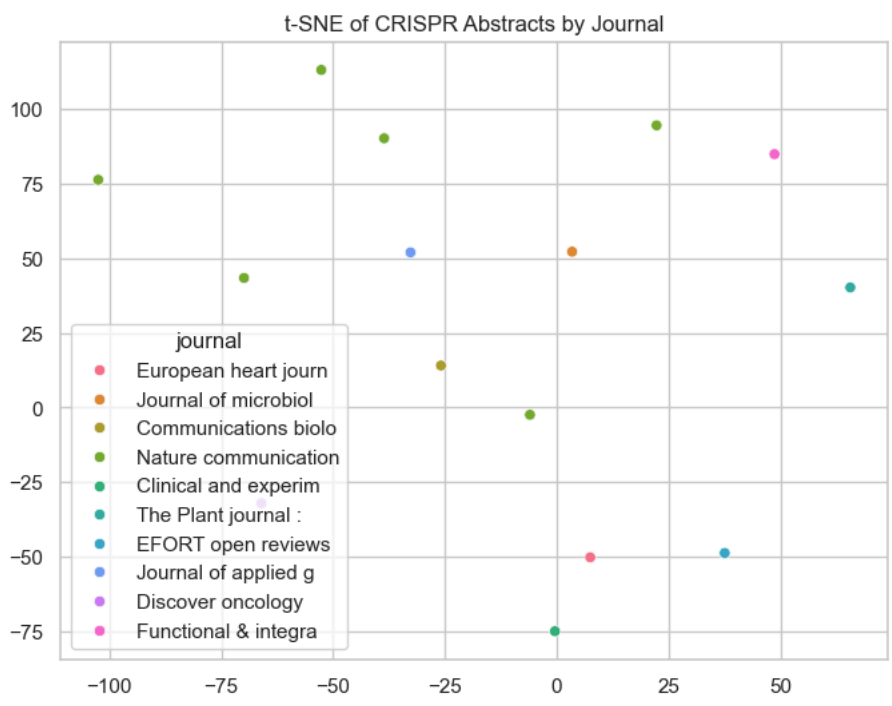
```
def extract_entities(text):
    doc = nlp(text)
    entities = []
    for ent in doc.ents:
```

```
        if ent.label_ in ["GENE", "PROTEIN", "CHEMICAL", "DISEASE"]:
            entities.append((ent.text, ent.label_))
    return entities

# Apply to first abstract
sample_abstract = crispr_df.iloc[0]["abstract"]
entities = extract_entities(sample_abstract)

# Visualize
doc = nlp(sample_abstract)
displacy.render(doc, style="ent", jupyter=True)
```

Genetic factors play a critical role in the development of lipoprotein disorders, which significantly contribute to atherosclerotic cardiovascular disease (ASCVD ORG). Traditional management of these conditions has relied on lipid-lowering therapies, which require lifelong adherence. Recent advancements in gene addition and editing technologies offer novel and potentially transformative approaches for treating lipoprotein disorders by targeting the relevant genetic pathways for each disease. This review revisits major monogenic ORG and polygenic disorders of lipoprotein metabolism, including familial hypercholesterolemia, elevated lipoprotein(a), and familial chylomicronemia syndrome, and discusses the genetic-based therapies for management. RNA ORG -based, gene addition and gene editing therapies, including Clustered Regularly Interspaced Short Palindromic Repeats, base editing and interventions whereby, are highlighted for their

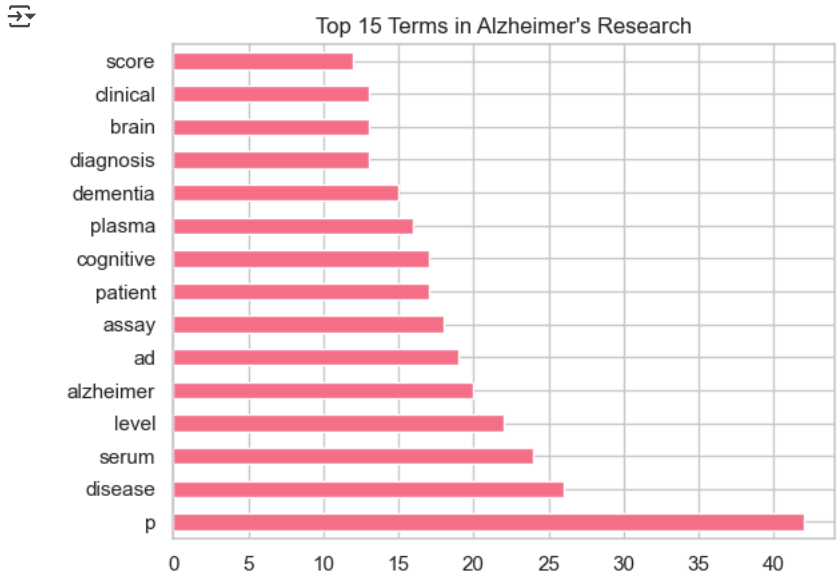
Dimensionality Reduction for Text Visualization

```
First, it uses a method called TF-IDF, which turns each abstract into a list of important words and how often they appear.
TF-IDF stands for:
    TF = Term Frequency: How often a word appears in a document.
    IDF = Inverse Document Frequency: How rare the word is across all documents.
It turns the text from each article into numbers using a method called TF-IDF, which finds the most important words in each summary.
Then it uses t-SNE, a tool that helps visualize those numbers in two dimensions, so we can plot them on a chart.
Finally, it creates a scatter plot, where each dot is a paper, and the color shows which journal it came from.

Need:
    Compare articles mathematically
    Group similar ones together
    Visualize them using tools like t-SNE


# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=100)
X = tfidf.fit_transform(crispr_df["processed_abstract"])

# t-SNE Visualization
tsne = TSNE(n_components=2, perplexity=5)
X_tsne = tsne.fit_transform(X.toarray())

# Plot
plt.figure(figsize=(8,6))
sns.scatterplot(x=X_tsne[:,0], y=X_tsne[:,1], hue=crispr_df["journal"].str[:20])
plt.title("t-SNE of CRISPR Abstracts by Journal");
```



Practical Exercise 1.Try fetching papers on "Alzheimer's disease biomarkers" 2.Compare word clouds between two search terms 3.Identify what biological entities are most mentioned in oncology papers

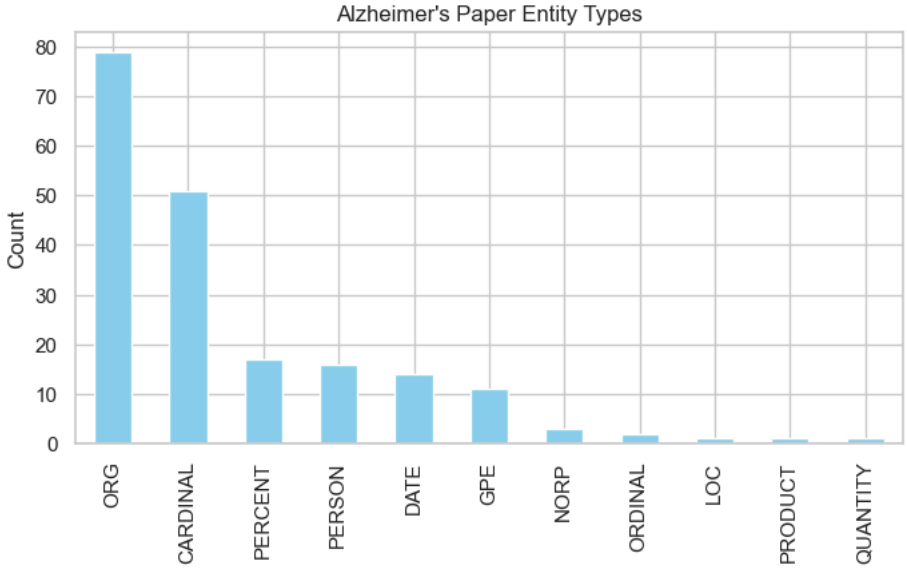
```
#1.Try fetching papers on "Alzheimer's disease biomarkers"
# Fetch Alzheimer's papers
alz_df = fetch_pubmed_abstracts("Alzheimer's disease biomarkers", 10)

# Process the text
alz_df["processed_abstract"] = alz_df["abstract"].apply(process_text)

# Show top terms
alz_word_freq = pd.Series(" ".join(alz_df["processed_abstract"]).split()).value_counts()[:15]
alz_word_freq.plot(kind="barh", title="Top 15 Terms in Alzheimer's Research");
```




```
#2.Compare word clouds between two search terms
# Get CRISPR data if not already loaded
crispr_df = fetch_pubmed_abstracts("CRISPR gene editing", 10)
crispr_df["processed_abstract"] = crispr_df["abstract"].apply(process_text)

# Create comparison figure
plt.figure(figsize=(15,6))

# Alzheimer's word cloud
plt.subplot(1,2,1)
alz_text = " ".join(alz_df["processed_abstract"])
alz_wc = WordCloud(width=400, height=300, background_color="white").generate(alz_text)
plt.imshow(alz_wc)
plt.title("Alzheimer's Biomarkers")
plt.axis("off")

# CRISPR word cloud
plt.subplot(1,2,2)
crispr_text = " ".join(crispr_df["processed_abstract"])
crispr_wc = WordCloud(width=400, height=300, background_color="white").generate(crispr_text)
plt.imshow(crispr_wc)
plt.title("CRISPR Research")
plt.axis("off")

plt.tight_layout()
plt.show()
```

This code looks through a collection of Alzheimer research abstracts and automatically identifies different types of important terms.
It goes through each abstract, picks out these named entities, and then counts how often each type appears.
Finally, it creates a bar chart to show which types of terms are most commonly mentioned in Alzheimer’s papers.
This gives us a quick overview of what kinds of biological information researchers are focusing on in this field.

```
#Combined Analysis
# Compare entity types between fields
alz_entities = []
for abstract in alz_df["abstract"]:
    doc = nlp(abstract)
    for ent in doc.ents:
        alz_entities.append(ent.label_)

plt.figure(figsize=(8,4))
pd.Series(alz_entities).value_counts().plot(kind="bar", color="skyblue")
plt.title("Alzheimer's Paper Entity Types")
plt.ylabel("Count");
```



Start coding or generate with AI.