## ****How to Interpret**** pyLDAvis ****LDA Visualization****

### ****1. Inter-topic Distance Map (Left Panel)****

**Circles Represent Topics:**  
Each circle corresponds to a topic found by the LDA model.

**Proximity = Similarity:**

**Close together** → Topics share many similar words.

**Far apart** → Topics have distinct vocabularies.

**Circle Size = Prevalence:**  
The area of each circle shows the proportion of the corpus that the topic represents.  
Larger circles = more dominant topics.

### ****2. Top-30 Most Relevant Terms for the Selected Topic (Right Panel)****

**Bar Chart Explanation:**

**Red bars** = Frequency of the term within the selected topic.

**Blue bars** = Overall frequency of the term in the entire dataset.

**Term Ranking:**  
Shows the **most relevant terms** for the selected topic.

**Relevance Metric (λ):**

Slider at the top adjusts the balance between:

**Topic-specific relevance** (lower λ, e.g., 0.2): highlights words unique to that topic.

**Overall frequency** (higher λ, e.g., 1.0): highlights common words that still appear in the topic.

**Best practice:** Start around λ = 0.6 for a balanced view.

### ****3. Identifying Overlapping Topics****

If circles **overlap**, it means topics share many terms, which can indicate:

Topics may be merged for simplicity.

The model might be using too many topics.

If circles are **well-separated**, it means topics are distinct and easy to interpret.

### ****4. Common Observations****

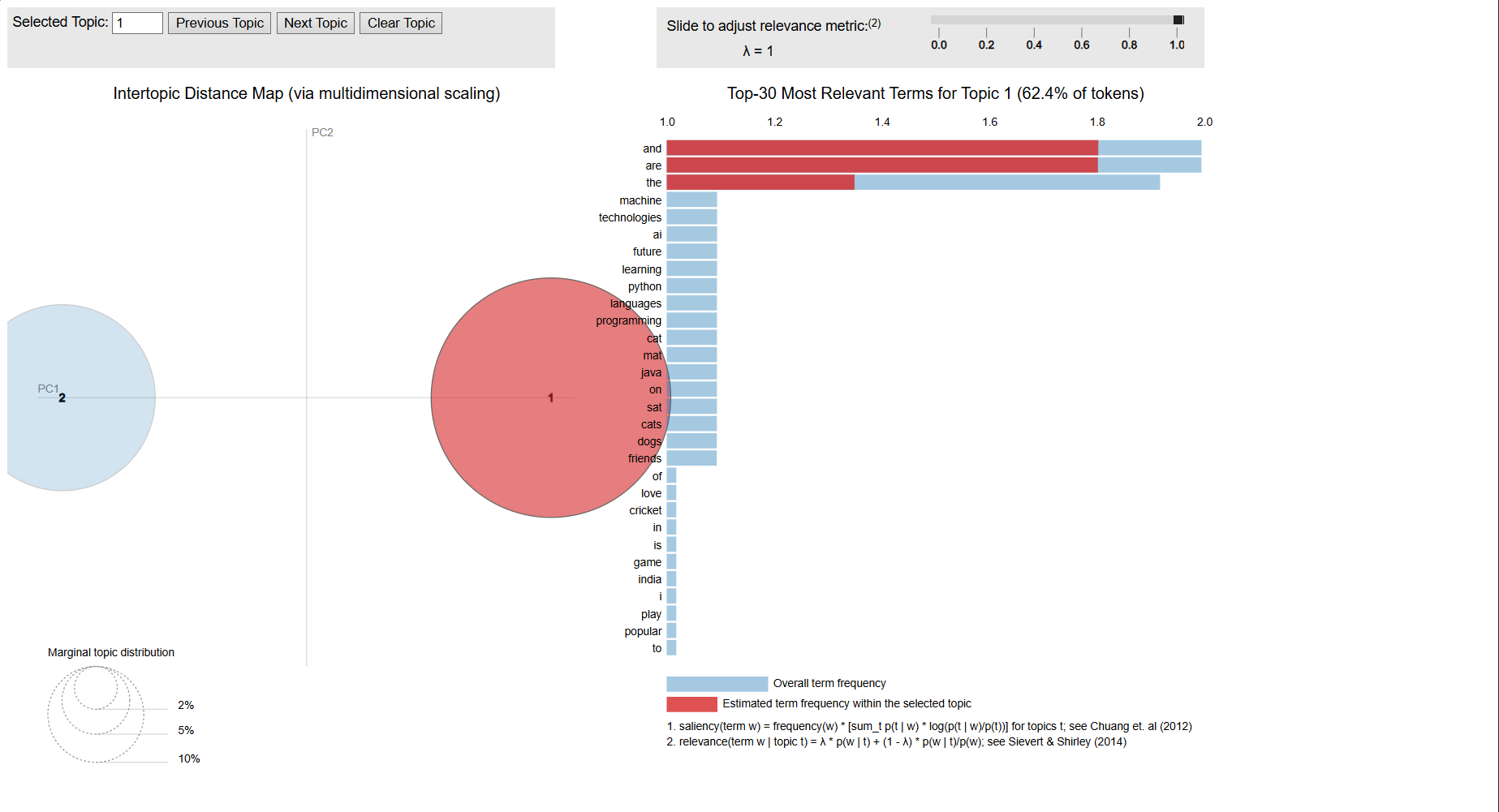
**Few Large Circles** → Corpus dominated by a few themes.

**Many Small Circles** → Corpus covers many smaller topics.

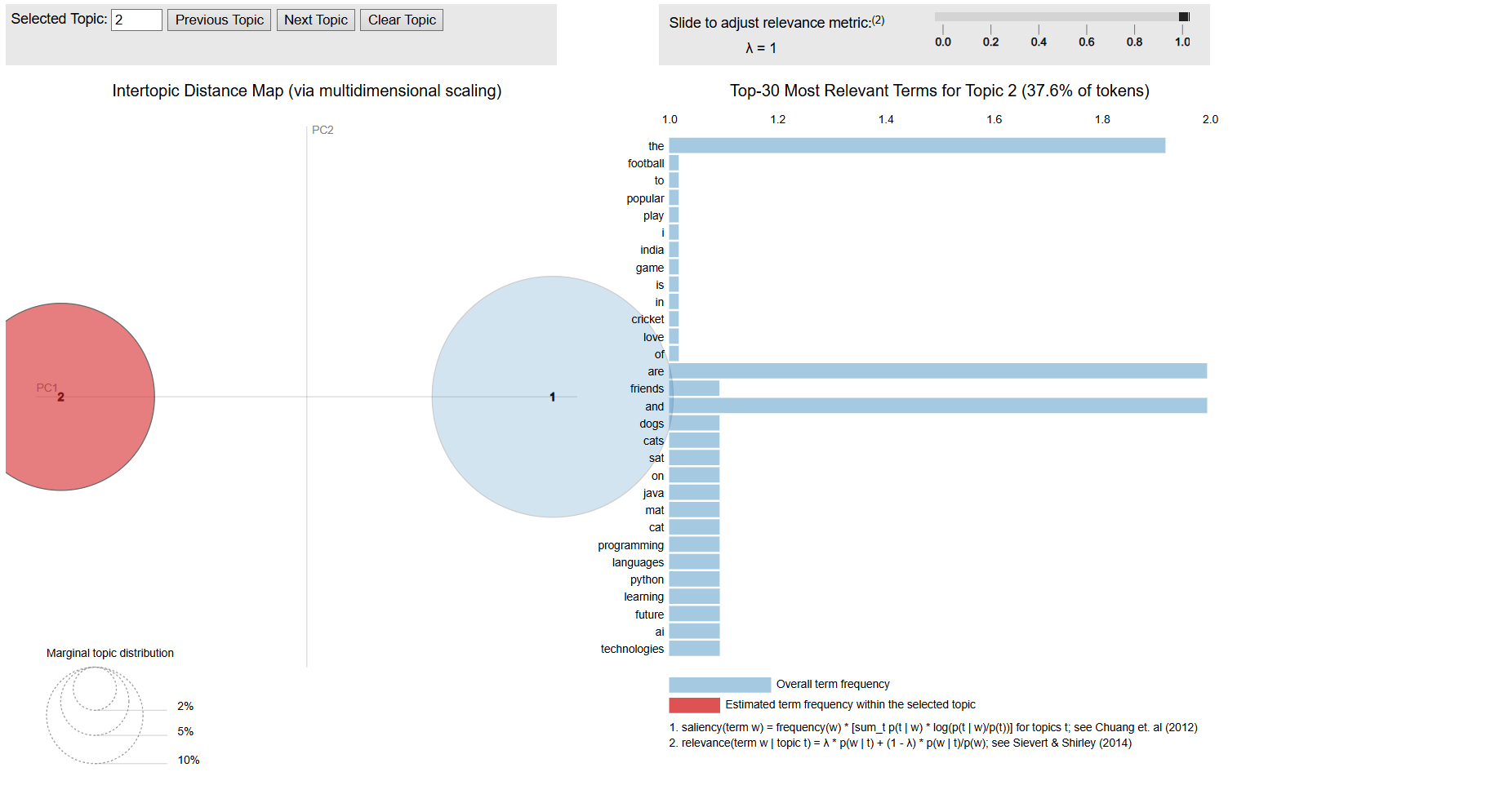
**Highly Overlapping Circles** → Need to reduce topic count or improve preprocessing (stopword removal, lemmatization, etc.).

**Examples**

1.



2.



## ****Python Code****

## ****1. Imports****

import osimport reimport stringfrom gensim import corpora, modelsfrom gensim.parsing.preprocessing import STOPWORDSfrom gensim.utils import simple\_preprocessfrom wordcloud import WordCloudimport matplotlib.pyplot as pltimport pyLDAvisimport pyLDAvis.gensim\_models as gensimvis

**os, re, string** → Handle file paths, regex cleanup, punctuation removal.

**gensim.corpora, models** → LDA implementation and corpus/dictionary creation.

**STOPWORDS** → Common words like “the”, “and”, “is” to remove.

**simple\_preprocess** → Gensim’s tokenizer that lowercases and removes non-alphabetic chars.

**WordCloud** → For generating topic-based word cloud images.

**matplotlib.pyplot** → For plotting and saving word clouds.

**pyLDAvis** → For interactive HTML visualization of topics.

## ****2. Configuration****

OUTPUT\_DIR = "lda\_outputs\_2topics"

HTML\_PATH = os.path.join(OUTPUT\_DIR, "topics\_2.html")

WC\_DIR = os.path.join(OUTPUT\_DIR, "wordclouds")

os.makedirs(WC\_DIR, exist\_ok=True)

Creates a folder structure so outputs don’t get messy.

HTML\_PATH → Path for the interactive topic visualization.

WC\_DIR → Path where **word cloud images** will be saved.

## ****3. Sample Text****

documents = [

"The cat sat on the mat!",

"Dogs and cats are friends.",

"I love to play football and watch cricket.",

"The game of cricket is popular in India!",

"Python and Java are programming languages.",

"Machine learning and AI are future technologies."

]

Just **example data** here — in real usage, replace with your text dataset.

A mix of sports, programming, and pets to produce at least **two clear topics**.

## ****4. Preprocessing****

def clean\_text(doc):

doc = doc.lower() # Lowercase

doc = doc.translate(str.maketrans("", "", string.punctuation)) # Remove punctuation

tokens = [t for t in simple\_preprocess(doc, deacc=True) if t not in STOPWORDS] # Tokenize + stopword removal

return tokens

texts = [clean\_text(doc) for doc in documents]

**Lowercase** → Makes “Cat” and “cat” the same.

**Remove punctuation** → So “cat,” becomes “cat”.

**Tokenization** → Breaks sentences into words.

**Stopword removal** → Removes common words like “the”, “is”, “and”.

texts becomes a list of cleaned word lists, e.g.:

[['cat', 'sat', 'mat'], ['dogs', 'cats', 'friends'], ...]

## ****5. Dictionary & Corpus****

dictionary = corpora.Dictionary(texts)

corpus = [dictionary.doc2bow(t) for t in texts]

**Dictionary** → Maps each word to an ID, e.g., { 'cat': 0, 'sat': 1, ... }.

**BOW (Bag of Words)** → Turns each document into word frequency counts.

Example: "cat sat mat" → [(0, 1), (1, 1), (2, 1)] meaning word ID 0 appears once, ID 1 appears once, etc.

## ****6. Train LDA****

num\_topics = 2

lda = models.LdaModel(

corpus=corpus,

id2word=dictionary,

num\_topics=num\_topics,

random\_state=42,

passes=10,

alpha="auto",

eta="auto",

)

**num\_topics = 2** → We only want **two themes**.

**passes=10** → How many times the model goes over the corpus to refine topics.

**alpha, eta = "auto"** → Lets Gensim tune topic distribution.

## ****7. Save Interactive HTML****

vis = gensimvis.prepare(lda, corpus, dictionary, mds='mmds')

html = pyLDAvis.prepared\_data\_to\_html(vis)with open(HTML\_PATH, "w", encoding="utf-8") as f:

f.write(html)

prepare() → Creates the interactive visualization data.

prepared\_data\_to\_html() → Converts it into a **standalone HTML file** (no external JS needed).

**Output**: A file you can open in a browser to explore topics visually.

## ****8. Generate Word Clouds****

for t in range(num\_topics):

freqs = dict(lda.show\_topic(t, topn=30))

wc = WordCloud(width=900, height=500, background\_color="white").generate\_from\_frequencies(freqs)

plt.imshow(wc, interpolation="bilinear")

plt.axis("off")

plt.tight\_layout()

plt.savefig(os.path.join(WC\_DIR, f"topic\_{t}\_k2.png"), dpi=150)

plt.close()

For **each topic**:

Get the top **30 words** and their probabilities.

Generate a **WordCloud** where word size = importance.

Save to wordclouds/topic\_X\_k2.png.