**Lesson 3: Topic Modeling and Text Summarization**

In this lesson, we’ll dive into two powerful unsupervised NLP techniques: Topic Modeling and Text Summarization. Both help us make sense of large volumes of text by extracting the most relevant information.

1. **What is Topic Modeling?**

**Topic Modeling** is an unsupervised technique that uncovers the hidden thematic structure in large text collections. It clusters words into "topics" and assigns documents to these topics based on word usage.

Common Algorithms:

* **LDA (Latent Dirichlet Allocation)** – the most widely used probabilistic model for topic discovery.
* **NMF (Non-negative Matrix Factorization)** – matrix factorization approach for identifying topics.

Example Use Cases:

* **Grouping customer reviews by underlying themes.**
* **Analyzing research papers for dominant topics.**
* **Organizing news articles.**

1. **Implementing LDA with Gensim**

**import gensim**

**from gensim import corpora**

**from gensim.models.ldamodel import LdaModel**

**from nltk.corpus import stopwords**

**from nltk.tokenize import word\_tokenize**

**import nltk**

**nltk.download('punkt')**

**nltk.download('stopwords')**

**# Sample Documents**

**documents = [**

**"Machine learning is a field of artificial intelligence.",**

**"Natural language processing involves text and speech.",**

**"Deep learning models are used for image and speech recognition.",**

**"Transformers have revolutionized NLP tasks like translation and summarization."**

**]**

**# Preprocessing**

**stop\_words = set(stopwords.words('english'))**

**texts = [**

**[word for word in word\_tokenize(doc.lower()) if word.isalpha() and word not in stop\_words]**

**for doc in documents**

**]**

**# Create Dictionary and Corpus**

**dictionary = corpora.Dictionary(texts)**

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

**# LDA Model**

**lda = LdaModel(corpus=corpus, id2word=dictionary, num\_topics=2, passes=10)**

**# Print Topics**

**for idx, topic in lda.print\_topics(-1):**

**print(f"Topic {idx}: {topic}")**

1. **Evaluating Topic Models**

* **Coherence Score: Measures how semantically related the words in a topic are.**
* **Perplexity: Lower values generally indicate a better model (used less frequently now).**
* **Code Example**

**from gensim.models.coherencemodel import CoherenceModel**

**coherence\_model = CoherenceModel(model=lda, texts=texts, dictionary=dictionary, coherence='c\_v')**

**print("Coherence Score:", coherence\_model.get\_coherence())**

1. **What is Text Summarization?**

**Text summarization is the process of distilling the most important information from a text.**

**Two Types:**

* **Extractive Summarization: Selects key sentences from the text (e.g., TextRank).**
* **Abstractive Summarization: Generates new sentences that summarize the content, like humans do (e.g., Transformers like BART, T5).**

1. **TextRank for Extractive Summarization (spaCy + NetworkX)**

**import spacy**

**import networkx as nx**

**from sklearn.metrics.pairwise import cosine\_similarity**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**import numpy as np**

**nlp = spacy.load("en\_core\_web\_sm")**

**text = """**

**Natural Language Processing (NLP) is transforming how we interact with machines. From chatbots to voice assistants, NLP helps machines understand human language. It’s a critical component of AI systems used in healthcare, finance, and customer service.**

**"""**

**# Sentence Tokenization**

**doc = nlp(text)**

**sentences = [sent.text.strip() for sent in doc.sents]**

**# TF-IDF Vectorization**

**vectorizer = TfidfVectorizer()**

**X = vectorizer.fit\_transform(sentences)**

**# Cosine Similarity Matrix**

**sim\_matrix = cosine\_similarity(X)**

**# Build Graph and Rank Sentences**

**nx\_graph = nx.from\_numpy\_array(sim\_matrix)**

**scores = nx.pagerank(nx\_graph)**

**# Rank Sentences**

**ranked\_sentences = sorted(((scores[i], s) for i, s in enumerate(sentences)), reverse=True)**

**# Summary (top 2 sentences)**

**summary = " ".join([s for \_, s in ranked\_sentences[:2]])**

**print("Summary:", summary)**

1. **Abstractive Summarization with Hugging Face Transformers**

**from transformers import pipeline**

**summarizer = pipeline("summarization")**

**text = """**

**The field of Natural Language Processing has seen tremendous growth. With the advent of transformer-based models like BERT and GPT, machines are now able to generate human-like text, translate languages, and answer questions more accurately than ever before.**

**"""**

**summary = summarizer(text, max\_length=50, min\_length=25, do\_sample=False)**

**print("Summary:", summary[0]['summary\_text'])**

1. **Comparison Table**

| **Technique** | **Type** | **Tools/Libraries** | **Best For** |
| --- | --- | --- | --- |
| **LDA** | **Unsupervised** | **Gensim** | **Discovering hidden themes in large corpora** |
| **TextRank** | **Extractive** | **spaCy, NetworkX, sklearn** | **Quick summaries from raw text** |
| **Transformer Summarizers** | **Abstractive** | **Hugging Face Transformers** | **High-quality, human-like summaries** |

1. **Summary**

* **Topic Modeling helps discover hidden themes in documents.**
* **LDA is a powerful unsupervised algorithm for topic discovery.**
* **Text Summarization can be extractive (TextRank) or abstractive (Transformers).**
* **spaCy, Gensim, NetworkX, and Hugging Face provide robust tools for these tasks.**