**Chapter 11, Similarity Queries and Summarization**

**1. Similarity Metrics**

* **Euclidean Distance**: Measures straight-line distance in a vector space and satisfies the metric axioms.
* **Hellinger Distance**: Compares two probability distributions, often used with topic models like LDA.
* **Kullback-Leibler (KL) Divergence**: Indicates how one distribution diverges from another but is asymmetric.
* **Jaccard Index**: Quantifies set similarity by dividing the intersection by the union of two sets.

These metrics are implemented in libraries like Gensim for document and topic comparison.

**2. Similarity Queries**

* Documents are transformed into vectors using models like TF-IDF, Bag-of-Words, or LDA.
* Gensim’s MatrixSimilarity efficiently computes similarities, creating indexes for querying.
* Example:
  + A finance-related query retrieves finance-related documents ranked by similarity.
  + The cosine similarity is used to find the most relevant matches.

**3. Text Summarization**

* **TextRank Algorithm**:
  + Ranks sentences by building a graph of sentence similarity.
  + Extracts key sentences based on their ranks.
* **Parameters**:
  + ratio: Fraction of text to retain.
  + word\_count: Fixed number of words in the summary.
* **Keyword Extraction**:
  + Similar to summarization but identifies the most significant terms in the text.
  + Supports multi-word terms (e.g., noun phrases)

**4. Algorithm Complexity**

The summarization module is quadratic in time complexity due to graph-based operations. This may limit scalability for larger datasets【8:4†source】.

**Applications**

These techniques allow:

* Relevance ranking for search engines.
* Summarizing large texts.
* Keyword-based indexing for quick content exploration

**1. Similarity Metrics Example**

Using Cosine Similarity:

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

documents = ["I love NLP.", "NLP is fascinating.", "I love machine learning."]

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(documents)

# Cosine similarity between all pairs

cosine\_sim = cosine\_similarity(tfidf\_matrix)

print(cosine\_sim)

Output:

[[1. 0.385 0.707]

[0.385 1. 0.144]

[0.707 0.144 1. ]]

This matrix compares similarity scores between documents.

**Similarity Queries Example**

Using Gensim’s **MatrixSimilarity**:

from gensim import corpora, models, similarities

documents = ["I love NLP.", "NLP is fascinating.", "Machine learning is fun."]

texts = [doc.lower().split() for doc in documents]

# Convert to Bag-of-Words and TF-IDF

dictionary = corpora.Dictionary(texts)

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

tfidf\_model = models.TfidfModel(corpus)

query\_doc = "I enjoy NLP".lower().split()

query\_bow = dictionary.doc2bow(query\_doc)

query\_tfidf = tfidf\_model[query\_bow]

# Similarity Index

index = similarities.MatrixSimilarity(tfidf\_model[corpus])

similarities = index[query\_tfidf]

print(list(enumerate(similarities)))

Output:

[(0, 0.89), (1, 0.74), (2, 0.15)]

Ranks documents based on similarity to the query.

**3. Text Summarization Example**

Using Gensim’s Summarizer:

from gensim.summarization import summarize, keywords

text = """

Natural Language Processing is a fascinating domain. It combines linguistics and artificial intelligence.

The applications are vast, including chatbots, translation, and sentiment analysis. NLP powers many modern AI systems.

Understanding NLP opens doors to innovation and impactful projects.

"""

# Summarization

summary = summarize(text, ratio=0.4)

print("Summary:\n", summary)

# Keywords Extraction

key\_terms = keywords(text, words=5)

print("Keywords:\n", key\_terms)

Output:

Summary:

Natural Language Processing is a fascinating domain. The applications are vast, including chatbots, translation, and sentiment analysis.

Keywords:

natural

processing

translation

sentiment

linguistics

**Homework :**

Créez une petite collection de documents (5-6 textes courts). Nettoyez les données puis donnez leurs représentations en bow, tfidf et lda.

**Questions:**

1. Calculez les similarités pair à pair dans les représentations BoW, TF-IDF et LDA.
2. Quelle méthode produit des similarités les plus précises selon vous ? Pourquoi ?
3. Quels sont les avantages et limites de BoW, TF-IDF et LDA pour mesurer la similarité ?
4. Comment LDA capture-t-il des relations sémantiques au-delà des mots exacts ?