Implementación de Latent Semantic Indexing en un Sistema de Recuperación de Información

Josue Rolando Naranjo Sieiro
Estudiante de 3er año, Ciencias de la Computación
Universidad de La Habana

Abril de 2025

Índice general

Capítulo 1

Introducción

La recuperación de información (RI) consiste en encontrar documentos relevantes a partir de consultas textuales. En modelos clásicos, la búsqueda literal de términos padece problemas de *sinonimia* y *polisemia*. Para mitigar estos efectos, implementamos el modelo *Latent Semantic Indexing* (LSI), que proyecta documentos y consultas a un espacio semántico de baja dimensión mediante descomposición en valores singulares (SVD), mejorando la recuperación basada en conceptos latentes.

Capítulo 2

Fundamentos de LSI

2.1. Definición formal

Sea

$$A \in R^{m \times n}$$

la matriz término-documento ponderada por TF-IDF. La descomposición SVD es

$$A = U \, \Sigma \, V^T,$$

y su truncación a rango k produce

$$A_k = U_k \, \Sigma_k \, V_k^T,$$

donde $U_k \in R^{m \times k}$, $\Sigma_k \in R^{k \times k}$ y $V_k \in R^{n \times k}$. Cada fila de $V_k \Sigma_k$ es el vector latente de un documento, y cada fila de $U_k \Sigma_k$ el vector latente de un término.

2.2. Ventajas

- Captura conceptos latentes y co-ocurrencias.
- Aumenta *recall* al encontrar sinónimos.
- Reduce ruido al truncar componentes de baja varianza.

Capítulo 3

Implementación

3.1. Estructura del proyecto

El repositorio contiene:

- `template.py`: clase base a personalizar.
- `main.py`: orquestador que evalúa todos los modelos.
- `metrics.py`: cálculos de precision, recall y f1.
- `ranking.py`: generación de tabla de resultados.
- `start.sh`: script de ejecución.

3.2. Preprocesamiento y vectorización

Configurar el vectorizador TF-IDF en `__init__`:

```
self.vectorizer = TfidfVectorizer(
    lowercase=True,
    stop_words='english',
    max_df=0.8
)
```

- **lowercase=True:** unifica mayúsculas/minúsculas.
- **stop_words='english':** filtra términos triviales.
- **max_df=0.8:** ignora términos muy frecuentes.

3.3. Construcción del espacio latente

En el método `fit`:

```
self.tfidf_matrix = self.vectorizer.fit_transform(self.documents)
self.svd = TruncatedSVD(n_components=100, random_state=42)
self.doc_latent_matrix = self.svd.fit_transform(self.tfidf_matrix)
```

- $\mathbf{A}$ = TF-IDF matriz.
- U_k, Σ_k, V_k obtenidos con `TruncatedSVD`.

3.4. Búsqueda y ranking

En `predict(top_k)`:

```
query_vec = self.vectorizer.transform([query_text])
query_latent = self.svd.transform(query_vec)
sim_scores = cosine_similarity(query_latent, self.doc_latent_matrix)[0]
top_idx = sim_scores.argsort()[::-1][:top_k]
retrieved_ids = [self.doc_ids[i] for i in top_idx]
return {'text': query_text, 'results': retrieved_ids}
```

3.5. Aserciones y validaciones internas

Para asegurar la consistencia:

```
assert self.tfidf_matrix.shape[0] == len(self.doc_ids)
assert self.doc_latent_matrix.shape == (len(self.doc_ids), self.svd.n_components_)
```

Capítulo 4

Pruebas y evaluación

4.1. Ejecución principal

```
bash start.sh
```

4.2. Script de prueba puntual

```
# test_lsi.py
from models.Josue_Rolando_Naranjo_Sieiro_C_311 import InformationRetrievalModel
from metrics import calculate_metrics
import ir_datasets

model = InformationRetrievalModel()
model.fit('cranfield')
preds = model.predict(5)

# Elegir primera consulta con qrels
dataset = ir_datasets.load('cranfield')
first_qid = next(q.query_id for q in dataset.qrels_iter())
retrieved = preds[first_qid]['results']
relevant = {qrel.doc_id for qrel in dataset.qrels_iter()
            if qrel.query_id == first_qid and qrel.relevance > 0}

scores = calculate_metrics(
    {'relevant_retrieved': set(retrieved)&relevant,
     'all_retrieved': set(retrieved),
     'all_relevant': relevant},
    ['precision','recall','f1']
)
print(first_qid, scores)
```

Capítulo 5

Información Extra

5.1. Fuente bibliográfica

- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). *Indexing by Latent Semantic Analysis*. Journal of the American Society for Information Science, 41(6): 391–407. `https://www.cs.csustan.edu/~mmartin/LDS/Deerwester-et-al.pdf`
- Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S. C. & Harshman, R. (1988). *Using latent semantic analysis to improve access to textual information*. SIGCHI Conference on Human Factors in Computing Systems.
- Berry, M. W., Dumais, S. T. & O'Brien, G. W. (1995). *Using linear algebra for intelligent information retrieval*. SIAM Review, 37(4): 573–595.
- Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Baeza-Yates, R. & Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology Behind Search* (2ª ed.). Addison-Wesley.

5.2. Mejora implementada

- **Técnica:** Configuración avanzada de TF-IDF (`lowercase=True`, `stop_words='english'`, `max_df=0.8`).
- **Beneficio:** Reducción de ruido y términos triviales, mejora de la calidad de los vectores latentes y de las métricas de precisión y recall.

5.3. Análisis conceptual

Definición formal: Descomposición SVD truncada de la matriz TF-IDF, $A_k = U_k \Sigma_k V_k^T$.

Dependencias entre términos: Sí. SVD captura patrones de co-ocurrencia, modelando relaciones semánticas.

Correspondencia parcial documento-consulta: Sí. Permite recuperar documentos sin coincidencia literal de términos.

Ranking: Sí. Documentos ordenados por similitud de coseno en el espacio latente.

Capítulo 6

Conclusión

Se ha diseñado e implementado un modelo LSI respetando las firmas de `fit` y `predict`, usando solo bibliotecas autorizadas. La configuración de preprocesamiento y la evaluación con métricas estándar garantizan un sistema robusto y extensible.