

Laboratorio 2 — Similitud de Coseno

Instrucciones

1. Carga de embeddings

- Utilizar las representaciones vectoriales preentrenadas (*GloVe 100d*).
- Verificar que los términos requeridos estén contenidos en el vocabulario.

2. Cálculo de vectores de expresión

- Construir expresiones vectoriales simples mediante operaciones aritméticas (suma, resta).

3. Medición de similitud

- Calcular la similitud de coseno entre los vectores de la expresión y los vectores de referencia.
- Reportar:
 - Valor de la similitud de coseno.
 - Normas de cada vector ($(\|\mathbf{u}\|)$).

4. Casos de estudio

Realizar caso por caso

5. Documentación de resultados

- Registrar en el notebook los valores obtenidos de similitud y normas vectoriales para cada caso.
- Incluir comentarios breves interpretando los resultados.

6. Conclusiones

- Redactar conclusiones **caso por caso** sobre la utilidad de la similitud de coseno.

```
In [ ]: # Solo si es necesario  
# !pip install gensim
```

Requirement already satisfied: gensim in c:\users\caste\anaconda3\lib\site-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in c:\users\caste\anaconda3\lib\site-packages (from gensim) (1.26.4)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in c:\users\caste\anaconda3\lib\site-packages (from gensim) (1.13.1)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\caste\anaconda3\lib\site-packages (from gensim) (5.2.1)

```
In [2]: import numpy as np
from typing import List
import gensim.downloader as api
```

```
In [3]: # ===== 1) Cargar GloVe 100D =====
model = api.load("glove-wiki-gigaword-100")
```

[=====] 100.0% 128.1/128.1MB downloaded

```
In [6]: # ===== 2) Funciones utilitarias =====
def embedding(word: str) -> np.ndarray:
    """Vector de la palabra (lanza KeyError si no está en vocab)."""
    return model[word]

def build_vector(plus: List[str], minus: List[str]) -> np.ndarray:
    """Suma y resta embeddings explícitamente."""
    dim = model[next(iter(model.key_to_index))].shape[0]
    v = np.zeros(dim, dtype=np.float32)
    for w in plus:
        if w in model:
            v += model[w]
        else:
            print(f"[OOV] '{w}' no está en el vocabulario.")
    for w in minus:
        if w in model:
            v -= model[w]
        else:
            print(f"[OOV] '{w}' no está en el vocabulario.")
    return v

def cos(a: np.ndarray, b: np.ndarray) -> float:
    na, nb = np.linalg.norm(a), np.linalg.norm(b)
    if na == 0 or nb == 0:
        return float("nan")
    return float(np.dot(a, b) / (na * nb))

def vec_norm(v: np.ndarray) -> float:
    return float(np.linalg.norm(v))

def report_case(label, plus, minus, target_plus):
    expr_vec = build_vector(plus, minus)
    target_vec = build_vector(target_plus, [])
    print("====", label, "====")
    print("Expresión =", " + ".join(plus) + " - " + " - ".join(minus) if minus else "")
    print("Target     =", " + ".join(target_plus))
    print("cos(expr, target) =", round(cos(expr_vec, target_vec), 6))
    print("||expr||      =", round(vec_norm(expr_vec), 6))
```

```

print("||target||      =", round(vec_norm(target_vec), 6))
print()

def nearest_neighbors(plus: List[str], minus: List[str], topn: int = 3):
    """Encuentra los vecinos más cercanos a una expresión vectorial."""
    expr_vec = build_vector(plus, minus)
    # calcular similitud con TODO el vocabulario
    sims = []
    for word in model.index_to_key:
        s = cos(expr_vec, model[word])
        if not np.isnan(s):
            sims.append((word, s))
    sims.sort(key=lambda x: -x[1])
    return sims[:topn]

```

Caso 1 — Capitales

Construya un vector que represente la relación:

paris + italy - france ≈ rome

1. Defina el vector de la expresión (`expr_vec`).
2. Defina el vector objetivo (`target_vec`).
3. Calcule `cos(expr, target)`, `||expr||` y `||target||`.
4. Escriba su conclusión.

```
In [7]: expr_vec = build_vector(["paris", "italy"], ["france"])
target_vec = build_vector(["rome"], [])
report_case("Caso 1", ["paris", "italy"], ["france"], ["rome"])
```

```
===== Caso 1 =====
Expresión = paris + italy - france
Target     = rome
cos(expr, target) = 0.8084
||expr||      = 6.227801
||target||     = 5.523363
```

La similitud de coseno obtenida fue 0.8084, lo cual indica una relación bastante fuerte entre el vector construido paris + italy - france y el vector de la palabra rome. Este resultado refleja que los embeddings de GloVe logran capturar correctamente la analogía capital-país, aproximando a Roma como capital de Italia al restar Francia y sumar Italia a París. Además, las normas de los vectores muestran que ambos tienen magnitudes consistentes, lo que da mayor validez a la comparación.

En conclusión, este experimento confirma la utilidad de los embeddings para modelar relaciones semánticas complejas a través de simples operaciones vectoriales. El hecho de que el coseno sea mayor a 0.8 demuestra que el modelo ha aprendido de los datos contextuales que Roma se asocia de manera muy cercana a Italia en la misma forma que París lo hace con Francia. Esto muestra cómo la similitud de coseno permite comprobar analogías semánticas de manera cuantitativa.

Caso 2 — Dictador/País

Construya un vector que represente la relación:

hitler + italy - germany

1. Construya el vector de la expresión.
2. Encuentre los **3 vecinos más cercanos** (`nearest_neighbors`).
3. Reporte la similitud coseno con cada vecino.
4. Escriba su interpretación de los resultados.

```
In [8]: expr_vec = build_vector(["hitler", "italy"], ["germany"])
neighbors = nearest_neighbors(["hitler", "italy"], ["germany"], topn=3)
print("Vecinos más cercanos (palabra, coseno):")
for w, s in neighbors:
    print(f" - {w:>15s} {s:.6f}")
```

Vecinos más cercanos (palabra, coseno):

- mussolini 0.816139
- hitler 0.715457
- fascist 0.668780

El vector `hitler + italy - germany` arrojó como vecino más cercano `mussolini` (0.8161), seguido por `hitler` (0.7155) y `fascist` (0.6688). Esto es coherente con la analogía buscada: al “mover” el concepto de Hitler desde Alemania hacia Italia, el espacio semántico se aproxima a Mussolini, líder fascista italiano, con una similitud de coseno alta (>0.81), lo que sugiere que GloVe capturó correctamente la relación “dictador \leftrightarrow país”.

Que `hitler` siga apareciendo entre los vecinos indica que la componente semántica de dictador/fascismo domina fuertemente el vector resultante, por lo que parte de la dirección original se conserva. La presencia de `fascist` como tercer vecino refuerza la dimensión ideológica compartida. En conjunto, los resultados respaldan la utilidad de la similitud de coseno para analogías históricas, aunque también evidencian limitaciones típicas de los embeddings (mezcla de rasgos ideológicos y de entidad) y posibles sesgos del corpus.

Caso 3 — Religión/Líder

Construya un vector que represente la relación:

christianity ≈ jesus

1. Construya el vector de la palabra `christianity`.
2. Encuentre los **3 vecinos más cercanos** (`nearest_neighbors`).
3. Reporte la similitud coseno con cada vecino.
4. Escriba su interpretación: ¿aparece `jesus` entre los vecinos?

```
In [9]: v_christ = build_vector(["christianity"], [])
neighbors = nearest_neighbors(["christianity"], [], topn=3)

print("Vecinos más cercanos (palabra, coseno):")
for w, s in neighbors:
    print(f" - {w:>15s} {s:.6f}")

cos_with_jesus = cos(v_christ, embedding("jesus")) if "jesus" in model else float("inf")
print("\n¿'jesus' está en top-3?:", any(w=="jesus" for w,_ in neighbors))
print("cos(christianity, jesus) =", f"{cos_with_jesus:.6f}")
print("||christianity|| =", f"{vec_norm(v_christ):.6f}")
print("||jesus||      =", f"{vec_norm(embedding('jesus')):.6f}" if "jesus" in model else float("inf"))
```

Vecinos más cercanos (palabra, coseno):

- christianity 1.000000
- catholicism 0.875112
- protestantism 0.811677

```
{'jesus' está en top-3?: False
cos(christianity, jesus) = 0.542634
||christianity|| = 5.906149
||jesus||      = 6.103833
```

El resultado muestra que los vecinos más cercanos a christianity son catholicism y protestantism, lo cual es coherente, ya que ambos son ramas principales de esta religión. Sin embargo, el término jesus no aparece entre los primeros vecinos y su similitud coseno con christianity es de apenas 0.54, un valor moderado que refleja cierta relación semántica, pero no lo suficientemente fuerte como para ubicarlo entre los más cercanos.

Esto indica que el modelo de embeddings representa la palabra christianity principalmente en términos de corrientes religiosas o denominaciones, más que en relación directa con su figura central. Aunque existe un vínculo vectorial con jesus, el contexto estadístico del corpus de entrenamiento de GloVe prioriza asociaciones doctrinales e institucionales. Este resultado evidencia tanto la potencia como las limitaciones de los embeddings al capturar diferentes dimensiones semánticas de un mismo concepto.

Caso 4 — Opuestos

Analice la relación:

good - bad

1. Construya el vector de la expresión `good - bad`.
2. Encuentre los **3 vecinos más cercanos** (`nearest_neighbors`).
3. Reporte la similitud coseno con cada vecino.
4. Escriba su conclusión sobre si los vecinos reflejan un contraste u oposición semántica.

```
In [10]: expr_vec = build_vector(["good"], ["bad"])
neighbors = nearest_neighbors(["good"], ["bad"], topn=3)
```

```

print("Vecinos más cercanos (palabra, coseno):")
for w, s in neighbors:
    print(f" - {w:>15s} {s:.6f}")
print("\nCos con 'good' : ", f"{cos(expr_vec, embedding('good')):.6f}")
print("Cos con 'bad'  : ", f"{cos(expr_vec, embedding('bad')):.6f}")
print("||expr||      : ", f"{vec_norm(expr_vec):.6f}")

```

Vecinos más cercanos (palabra, coseno):

- excellent 0.522756
- versatile 0.459313
- ideal 0.457311

Cos con 'good' : 0.400063

Cos con 'bad' : -0.276289

||expr|| : 3.815561

El vector good - bad produjo vecinos positive-oriented: excellent (0.5228), versatile (0.4593) e ideal (0.4573). Además, su proyección tiene coseno positivo con "good" (0.4001) y negativo con "bad" (-0.2763), lo que indica que la dirección del vector se aleja de lo negativo y se acerca a términos asociados con cualidades deseables. Esto sugiere que la operación efectivamente empuja la representación hacia un eje de "positividad".

Sin embargo, las similitudes son moderadas, no cercanas a 1, lo que evidencia una limitación conocida: los antónimos suelen compartir contextos y, por tanto, pueden quedar relativamente próximos en embeddings distribucionales. En síntesis, good - bad sí refleja contraste semántico (señal clara por el coseno negativo con bad), pero no define un "eje de oposición" perfectamente limpio; para capturar mejor se requerirían técnicas como retrofitting/counter-fitting con lexicones de sinonimia/antónimo.

Caso 5 — Propio

Defina usted mismo un caso interesante. Ejemplos:

- king - man + woman
- tokyo + france - japan
- apple - technology

1. Construya la expresión vectorial que haya definido.
2. Encuentre los **3 vecinos más cercanos** (`nearest_neighbors`).
3. Reporte la similitud coseno con cada vecino.
4. Redacte una conclusión explicando si los resultados corresponden a su hipótesis inicial.

```

In [11]: expr_vec = build_vector(["twitter", "photos"], ["text"])
neighbors = nearest_neighbors(["twitter", "photos"], ["text"], topn=5)

print("Vecinos más cercanos (palabra, coseno):")
for w, s in neighbors:
    print(f" - {w:>15s} {s:.6f}")

```

```

target = "instagram" if "instagram" in model else "flickr"
print(f"\ncos(expr, '{target}') =", f"{cos(expr_vec, embedding(target)):.6f}")
print("||expr||", " =", f"{vec_norm(expr_vec):.6f}")
if target in model:
    print(f"||'{target}'||", " =", f"{vec_norm(embedding(target)):.6f}")

```

Vecinos más cercanos (palabra, coseno):

- facebook 0.669745
- flickr 0.665407
- twitter 0.664536
- myspace 0.663974
- youtube 0.631841

```

cos(expr, 'instagram') = 0.597748
||expr||                 = 8.155613
||'instagram'||          = 5.159590

```

La expresión twitter + photos – text se desplazó hacia el eje de “redes sociales con fuerte componente visual”: los vecinos fueron facebook (0.6697), flickr (0.6654), twitter (0.6645), myspace (0.6640) y youtube (0.6318), y la similitud con instagram fue 0.5977. Esto sugiere que el vector capturó bien la idea de “plataforma social + énfasis visual”, reflejado en la cercanía a flickr y youtube, mientras aún conserva parte de la identidad de twitter por aparecer entre los vecinos.

Que instagram no lidere, pese a un coseno razonable (~0.6), muestra una limitación típica de GloVe: las marcas comparten mucho contexto (“social media”), por lo que el vector mezcla rasgos de varias plataformas. Además, el corpus es antiguo y flickr/mspace pesan más históricamente para “fotos”. Aun así, el resultado apoya parcialmente la hipótesis: el modelo se orienta a “social + visual”, aunque no distingue con precisión fina entre plataformas; para afinar, podrían usarse expresiones más específicas (“photo_sharing”), modelos con subpalabras (fastText) o embeddings contextuales.