Universidad del Valle de Guatemala
Deep Learning y Sistemas Inteligentes
Sección 10

Laboratorio #8

Embeddings

21016 Javier Chavez
21085 Andres Quezada

Práctica

1. Emplee la función Word2Vec de la librería gensim para generar embeddings a partir de los documentos del dataset. Elija los mejores valores a su opinion para cada uno de los parametros de la funcion (vector_size, window, min_count, workers).

   Hicimos un pre procesamiento de los datos, quitando los caracteres especiales como la puntuación y también investigamos que para mejorar los resultados se pueden eliminar palabras “vacías” como conectores como “and”, y esto da mejores resultados en los embeddings, ya que reduce el ruido.

   Elegimos:

   - vector_size=100: Elegimos 100 dimensiones para tener tanto precisión como un tiempo de ejecución decente.
   - window=5: Un tamaño de ventana de 5 palabras para capturar el contexto cercano.
   - min_count=2: Ignoramos palabras que aparecen solo una vez para reducir ruido.
   - workers=10: Utilizamos 10 núcleos para acelerar el entrenamiento.

2. Usando los embeddings generados, aplica la función most_similar para encontrar las 10 palabras más similares a cada una de las siguientes:

   • Street
   • Good
   • Dog
   • Mother
   • Bed

   Resultado:

   Palabras similares a 'street':
   road: 0.8793
   streetthe: 0.8392
   steet: 0.8368

streetwe: 0.8208
roadthe: 0.7958
stree: 0.7928
streeti: 0.7865
streetit: 0.7727
intersection: 0.7676
streeta: 0.7505

Palabras similares a 'good':
decent: 0.8450
great: 0.8207
excellent: 0.7793
terrific: 0.6776
exellent: 0.6669
excellant: 0.6519
reasonable: 0.6512
soso: 0.6433
goodthe: 0.6418
excelent: 0.6416

Palabras similares a 'dog':
dogs: 0.8193
kid: 0.5708
pet: 0.5687
babies: 0.5609
babys: 0.5434
pets: 0.5372
kids: 0.5301
inflatable: 0.5240
child: 0.5211
children: 0.5145

Palabras similares a 'mother':
father: 0.8791
daughter: 0.8608
mom: 0.8432
wife: 0.8387
husband: 0.8325
mum: 0.8294
brother: 0.8247
son: 0.8228

cousin: 0.8214
daughters: 0.8207

Palabras similares a 'bed':
beds: 0.8869
bedthe: 0.8618
bedi: 0.7654
daybed: 0.7610
bedwe: 0.7289
sofabed: 0.7217
bedsthe: 0.7040
bedthat: 0.6985
hideabed: 0.6837
mattress: 0.6821

3. Extrae los embeddings de estas 55 palabras (las 5 palabras iniciales más sus 10 palabras similares cada una).

   El contenido completo está en el notebook, este es un ejemplo del output:

   Palabra: street, Embedding: [3.760018 2.1485655 -3.0300589 2.0634162 1.9304447]... (truncated)
   Palabra: road, Embedding: [2.8664591 0.12245713 -2.4775164 2.880837 0.88593334]... (truncated)
   Palabra: streetthe, Embedding: [2.6822388 2.2242584 -2.430243 3.0168436 2.365246]... (truncated)
   Palabra: steet, Embedding: [1.1079963 0.43831077 -0.45958522 1.1204469 0.32989192]... (truncated)
   Palabra: streetwe, Embedding: [1.5599853 1.0685058 -1.8893888 0.56750023 0.7729391]... (truncated)

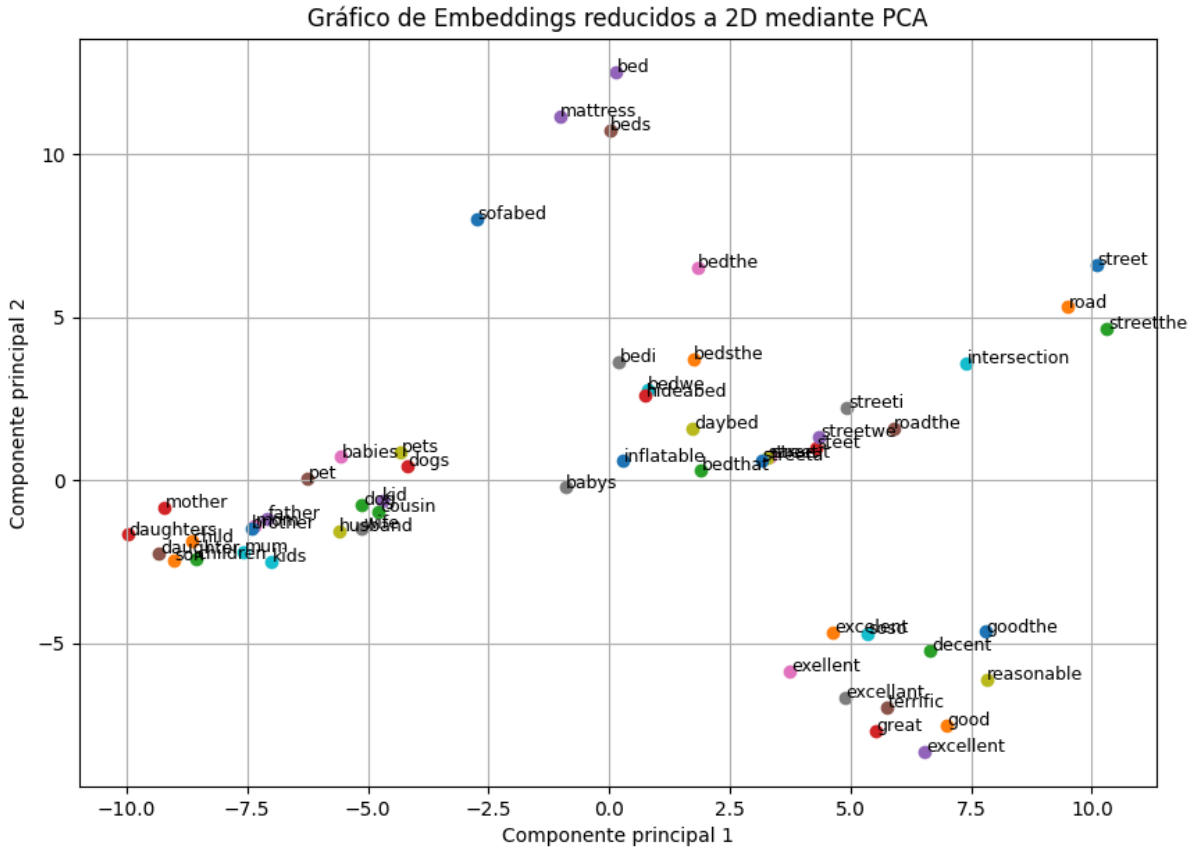
4. Aplica PCA para reducir la dimensionalidad de los embeddings a 2 componentes principales.

   Igual en el notebook se puede ver, aquí hay un ejemplo:

   Palabra: street, PCA1: 10.097508430480957, PCA2: 6.599677085876465
   Palabra: road, PCA1: 9.4938383102417, PCA2: 5.3011860847473145
   Palabra: streetthe, PCA1: 10.315581321716309, PCA2: 4.643979549407959
   Palabra: steet, PCA1: 4.2848968505859375, PCA2: 0.9900001287460327

Palabra: streetwe, PCA1: 4.352272987365723, PCA2: 1.3233652114868164
Palabra: roadthe, PCA1: 5.887058258056641, PCA2: 1.5777969360351562

5. Crea un gráfico de dispersión (scatter plot) con estos 2 componentes y describe los resultados obtenidos.



El gráfico de dispersión muestra la relación entre las 55 palabras proyectadas en un plano 2D. Palabras cercanas en el gráfico comparten un significado semántico similar según los embeddings generados. Las palabras que aparecen más alejadas en el gráfico tienen contextos menos relacionados en los textos analizados. Por eso podemos ver palabras como mother, father y daughter cercanas, o good, excelent, decent, etc.

Teoría

El constructor del modelo Word2Vec acepta dos parámetros importantes:

a. vector_size
b. window

Describa la función que cumple cada uno de estos parámetros dentro del algoritmo y discuta sobre las consecuencias de utilizar valores muy altos o muy bajos para cada uno de estos parámetros. Indique en qué situaciones podría ser apropiado ajustar de determinada manera (alto o bajo) estos valores o si, en ciertos casos, nunca es recomendable hacerlo.

- El parámetro vector_size define la cantidad de dimensiones del espacio donde se representan las palabras. Un valor alto permite captar más detalles y relaciones complejas entre palabras, pero hace que el modelo sea más lento y consuma más memoria. Un valor bajo entrena más rápido, pero los embeddings pueden no capturar tanta información. Es recomendable usar un valor alto cuando tenemos muchos datos, y un valor bajo si trabajamos con un dataset pequeño o si buscamos eficiencia.

- El parámetro window establece el tamaño del contexto, es decir, cuántas palabras cercanas se consideran para entender el significado de una palabra. Un valor alto captura relaciones más amplias entre palabras que están lejos entre sí, mientras que un valor bajo se enfoca en el contexto inmediato. Un valor alto puede ser útil en textos donde las relaciones a largo plazo son importantes, como artículos extensos, mientras que un valor bajo es útil en frases cortas o cuando el significado depende del contexto cercano.

- Ajustar estos valores depende del tipo de texto que estés procesando y del balance que busques entre precisión y tiempo de procesamiento. Nunca es recomendable usar valores extremadamente altos si los datos no justifican esa complejidad, ya que podrías sobrecargar el modelo sin ganar precisión.

Repositorio:

https://github.com/JaniMariQuesiRami/DeLe-Lab8