Word2Vec

1. Introducción

En esta práctica usamos el modelo Word2Vec con la librería Gensim. Lo que se busca es entender cómo se pueden transformar palabras en vectores que reflejan cierta relación semántica, usando ejemplos básicos. El enfoque fue más práctico que teórico para ir viendo directamente cómo funciona el modelo.

Carga del dataset

```
CopiarEditar
# Load the dataset from a CSV file
df = pd.read_csv('reddit_worldnews_start_to_2016-11-22.csv')

# Display the first few rows of the dataframe to understand its structure
df.head()
```

Se cargan los datos desde un archivo CSV. Este contiene noticias de Reddit en formato de tabla. Con `df.head()` vemos cómo vienen organizados los datos.

Descarga de recursos de NLTK

```
CopiarEditar
# Download the 'punkt' tokenizer models from NLTK
# Uncomment the lines below to download the models if not already downloaded
# nltk.download('punkt')
# nltk.download('punkt_tab')
```

Acá se mencionan los recursos que se deben descargar si no se tienen, para poder tokenizar los textos.

Tokenización de los títulos

```
CopiarEditar
# Tokenize the titles in the dataframe using NLTK's word_tokenize function
newsVec = [nltk.word_tokenize(title) for title in df['title'].values]

# Display the first 5 tokenized titles to verify the result
newsVec[:5]
```

Convertimos los títulos de noticias en listas de palabras (tokens). Esto es necesario para que Word2Vec pueda trabajar con los textos.

```
[['Scores', 'killed', 'in', 'Pakistan', 'clashes'],
 ['Japan', 'resumes', 'refuelling', 'mission'],
 ['US', 'presses', 'Egypt', 'on', 'Gaza', 'border'],
 ['Jump-start', 'economy', ':', 'Give', 'health', 'care', 'to', 'all'],
 ['Council', 'of', 'Europe', 'bashes', 'EU', '&', 'UN', 'terror', 'blacklist']]
```

Entrenamiento del modelo Word2Vec

```
CopiarEditar
# Train a Word2Vec model using the tokenized titles (newsVec)
# min_count=1 ensures that even words that appear only once are included in the
model
model = Word2Vec(newsVec, min_count=1).wv
```

Acá se entrena el modelo de Word2Vec desde cero usando los títulos tokenizados. Con `min_count=1` incluimos todas las palabras, incluso las raras.

Búsqueda de palabras similares a 'man'

```
CopiarEditar
# Find the most similar words to 'man' using the trained Word2Vec model
similar_words = model.most_similar('man')

# Display the similar words
similar_words
```

Buscamos las palabras más parecidas a "man" según lo que aprendió el modelo. Word2Vec representa palabras en vectores, y puede calcular su cercanía.

```
[('woman', 0.9072763919830322),
 ('teenager', 0.8432095646858215),
 ('boy', 0.8381759524345398),
 ('girl', 0.824225127696991),
 ('couple', 0.7924955487251282),
 ('teen', 0.7605805397033691),
 ('mother', 0.7585114240646362),
 ('policeman', 0.7574959397315979),
 ('doctor', 0.7536249160766602),
 ('teacher', 0.7390760779380798)]
```

Cálculo de similitud con suma de vectores

```
CopiarEditar
# Create a vector by adding the vectors for 'holiday', 'gifts', and 'winter'
vec = model['holiday'] + model['gifts'] + model['winter']

# Find the most similar words to the created vector
similar_words = model.most_similar([vec])

# Display the similar words
similar_words
```

Creamos un vector combinando palabras relacionadas a fiestas. Luego buscamos qué otras palabras son similares al resultado de esa suma.

```
[('holiday', 0.8276294469833374),
 ('holidays', 0.7679221630096436),
 ('gifts', 0.7360822558403015),
 ('Christmas', 0.7285211682319641),
 ('winter', 0.6602602005004883),
 ('festive', 0.6454193592071533),
 ('Thanksgiving', 0.6245847940444946),
 ('gift', 0.612133264541626),
 ('Christmastime', 0.6087110042572021),
 ('vacations', 0.5980472564697266)]
```

Uso de modelo pre-entrenado (Google News)

Antes de usar el modelo, se descarga de este enlace y se descomprime en la raíz del proyecto: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?resourcekey=0-wjGZdNAUop6WykTtMip30g

```
CopiarEditar
# Load the pre-trained Word2Vec model from the Google News dataset
# The model is in binary format, so we set binary=True
model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin',
binary=True)
```

Cargamos un modelo ya entrenado por Google. Este es mucho más robusto y fue entrenado con millones de noticias.

Operaciones con vectores del modelo preentrenado

```
CopiarEditar
# Create a vector by adding the vectors for 'king' and 'woman' and subtracting
the vector for 'man'
vec = model['king'] + model['woman'] - model['man']

# Find the most similar words to the created vector
similar_words = model.most_similar([vec])

# Display the similar words
similar_words
```

Un clásico ejemplo de Word2Vec. Si a "king" le sumamos "woman" y le restamos "man", el resultado se acerca a "queen".

```
[('king', 0.52085942029953),
 ('woman', 0.5135486721992493),
 ('monarch', 0.48635631799697876),
 ('crown_prince', 0.47217562794685364),
 ('prince', 0.4661101698875427),
 ('princess', 0.45525479316711426),
 ('man', 0.4482707381248474),
 ('teenage_girl', 0.4421442151069641),
 ('girl', 0.42170172929763794),
 ('boy', 0.40749162435531616)]
```

Conclusión

En esta práctica aprendimos cómo funciona Word2Vec y cómo usarlo con textos reales. Probamos tanto un modelo entrenado desde cero con noticias de Reddit, como uno preentrenado de Google. Esto permite explorar relaciones semánticas entre palabras de forma muy intuitiva.