TEMA 3 – HOJA DE EJERCICIOS V

Se plantean diversos ejercicios relacionados con el empleo de modelos preentrenados basados en la arquitectura de Word2Vec.

Ejercicio 1. Sense2Vec

Una de las principales limitaciones presentes en la implementación original de Word2Vec es la asignación de un único vector estático a cada palabra. Además de los posibles problemas relacionados con las palabras fuera del vocabulario de entrenamiento, este hecho puede suponer un gran inconveniente a la hora de analizar palabras polisémicas con diversas categorías gramaticales, no pudiendo diferenciar sus posibles acepciones.

En este contexto, Sense2Vec surge como una posible solución a este problema. En lugar de tratar cada token como una entidad única, Sense2Vec propone realizar un etiquetado previo de los tokens de entrenamiento con su correspondiente etiqueta gramatical o etiqueta Part-of-Speech (POS). Por medio de la incorporación de esta información lingüística, Sense2Vec permite generar representaciones diferenciadas de palabras según su categoría gramatical, facilitando el análisis de palabras polisémicas.

Para trabajar con Sense2Vec hay que descargar el modelo a partir del siguiente enlace: https://github.com/explosion/sense2vec/releases/download/v1.0.0/s2v_reddit_2015_md.tar.gz
Nota: si se trabaja desde Google Colab, puede ser interesante descargar el modelo mediante los siguientes comandos:

```
!wget
https://github.com/explosion/sense2vec/releases/download/v1.0.0/s2v_reddit_2015_md.tar.gz
!tar -xvzf s2v_reddit_2015_md.tar.gz
```

Una vez extraído el modelo de Sense2Vec "s2v_old", cárgalo mediante el siguiente código de ejemplo. Nota que <TU_PATH> debe reemplazarse por la ubicación donde se encuentra el modelo en tu sistema de archivos:

```
from sense2vec import Sense2Vec
s2v = Sense2Vec().from_disk("<TU_PAH>/s2v_old")
```

Se pide:

- Calcula con Sense2Vec los embeddings de la palabra “watch” tanto cuando ejerce la categoría gramatical de verbo como cuando ejerce la categoría gramatical de nombre. ¿Son diferentes los vectores?
- Dado el conjunto de frases *["You must watch the light movie.", "He gave me a watch with a light."]*, elimina las palabras vacías (stopwords) y signos de puntuación. Posteriormente, obtén con Sense2Vec las tres palabras más similares de los tokens resultantes, empleando para ello su categoría gramatical.
- Usando la librería Gensim, repite lo anterior para el modelo preentrenado de Word2Vec "word2vec-google-news-300". ¿Qué diferencias encuentras entre Word2Vec y Sense2Vec? ¿Cuál funciona mejor y sobre qué contextos?
- Con Word2Vec y Sense2Vec, repite lo anterior para el conjunto de frases ["I went to the bank to deposit my money.", "The land along the river bank has vegetation."]. ¿Qué ocurre en esta ocasión? ¿A qué se debe?
- Calcula la similitud coseno entre los siguientes pares de palabras con la categoría gramatical indicada:
  1. *("watch|NOUN", "watch|VERB")*
  2. *("watch|NOUN", "clock|NOUN")*
  3. *("watch|NOUN", "view|VERB")*
  4. *("watch|VERB", "clock|NOUN")*
  5. *("watch|VERB", "view|VERB")*

  ¿Es alto el valor de similitud coseno entre ("watch|NOUN", "watch|VERB")? ¿A qué podría deberse?

  Nota:

  - Se recomienda emplear Google Colab en caso de disponer de poca memoria RAM https://colab.research.google.com/
  - https://github.com/explosion/sense2vec

Ejercicio 2. Doc2Vec

Los modelos de embedding estáticos anteriormente vistos (Word2Vec, GloVe, FastText y Sense2Vec) generan representaciones a nivel de palabra. Sin embargo, a la hora de llevar a cabo diferentes tareas de procesamiento del lenguaje natural, podría ser interesante obtener una representación a nivel de documento que sea capaz de capturar la semántica global del mismo.

En este contexto, surge Doc2Vec, extensión de Word2Vec diseñada para representar textos de forma integral. A diferencia de los anteriores modelos preentrenados que únicamente generan embeddings a nivel de token, Doc2Vec introduce un vector que representa al documento en su totalidad. Con dicho fin, hace uso de o bien la técnica Distributed Memory (PV-DM), análoga a CBOW, o bien de la técnica Distributed Bag of Words (PV-DBOW), análoga a Skip-Gram. Específicamente, PV-DM predice una palabra en función de su contexto y del vector del documento, mientras que PV-DBOW omite el contexto y utiliza exclusivamente el vector para predecir las palabras que lo conforman.

Se pide lo siguiente:

- Prepara un conjunto aleatorio de 10000 resúmenes de Wikipedia en inglés. Para ello, puedes emplear el siguiente código de ejemplo, el cual usa la librería Datasets:

```
from datasets import load_dataset


dataset = load_dataset("laion/Wikipedia-Abstract", "English")
subset = dataset["train"].select(range(10000))
texts, titles = subset["Abstract"], subset["Title"]
```

- Utiliza TaggedDocument de la librería Gensim para preparar el dataset de entrenamiento. Tras ello, entrena un modelo Doc2Vec sobre el anterior conjunto de datos.

- Dados los resúmenes de Wikipedia de "Federico García Lorca", "Flag of Europe" y "Super Mario 64", calcula para cada uno sus 10 resúmenes más similares. Analiza los resultados. ¿Están relacionados?

- Dado el siguiente nuevo resumen no visto durante la etapa de entrenamiento, calcula su vector y analiza los resúmenes más similares al mismo:

Mycology is the branch of biology concerned with the study of fungi, including their taxonomy, genetics, biochemical properties, and use by humans.[1] Fungi can be a source of tinder, food, traditional medicine, as well as entheogens, poison, and infection. Yeasts are among the most heavily utilized members of the Kingdom Fungi, particularly in food manufacturing.[2]

Mycology branches into the field of phytopathology, the study of plant diseases. The two disciplines are closely related, because the vast majority of plant pathogens are fungi. A biologist specializing in mycology is called a mycologist

- Representa gráficamente con PCA y la librería interactiva Bokeh un subconjunto de los anteriores resúmenes de Wikipedia. Para ello, puedes emplear el siguiente código de ejemplo:

```
from bokeh.models import ColumnDataSource
from bokeh.plotting import figure, show
from sklearn.decomposition import PCA

selection_subset = subset.select(range(500))
texts, titles = selection_subset["Abstract"], selection_subset["Title"]

doc_vectors = [
        model.infer_vector(word_tokenize(text))
        for text in texts
]

pca = PCA(n_components=2)
vectors_2d = pca.fit_transform(doc_vectors)

source = ColumnDataSource({'x': vectors_2d[:, 0], 'y': vectors_2d[:, 1],     'title': titles})

p = figure(tools="pan,wheel_zoom,reset", width=1700, height=1000)
p.scatter(x='x', y='y', size=10, source=source, fill_alpha=0.6)
p.text(x='x', y='y', text='title', source=source, x_offset=5, y_offset=5)
show(p)
```

¿Qué resúmenes se encuentran más cerca? ¿Tiene sentido su cercanía?

- Sobre los apartados anteriores, experimenta diferentes configuraciones de entrenamiento de Doc2Vec (número de dimensiones, ventana de contexto, épocas, dm=0 o dm=1, ...)

Nota:

- https://huggingface.co/docs/datasets/index
- https://huggingface.co/datasets/laion/Wikipedia-Abstract
- https://radimrehurek.com/gensim/models/doc2vec.html
- https://bokeh.org/