Desafío - Inferencia de tópicos con EM

- Para realizar este desafío debes haber estudiado previamente todo el material disponibilizado correspondiente a la unidad.
- Una vez terminado el desafío, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el `.zip` en el LMS.
- Desarrollo desafío:
    - El desafío se debe desarrollar de manera Individual.
    - Para la realización del desafío necesitarás apoyarte del archivo *Apoyo Desafío - Inferencia de tópicos con EM*.

Requerimientos

- En esta sesión trabajaremos con una serie de base de datos sobre letras musicales de distintos artistas. Cada uno de los `csv` se encuentra en la carpeta `dump`.

- Cada `csv` tiene el nombre del artista a analizar. Los archivos contienen el nombre del artista, el género musical del artista, el nombre de la canción y las letras.

- En base a esta información, el objetivo del ejercicio es generar un modelo probabilístico que pueda identificar el género musical más probable dado la letra de una canción.

- Para ello implementaremos un modelo conocido como Latent Dirichlet Allocation que hace uso de una variante del algoritmo EM para inferir clases latentes a partir de una matriz de documentos.

Ejercicio 1: Preparar el ambiente de trabajo

- Importe los módulos `numpy`, `pandas`, `matplotlib`, `seaborn`, `glob` y `os` siguiendo las buenas prácticas. Los últimos dos módulos permitirán realizar la importación de múltiples archivos dentro de la carpeta `dump`.

- Para ello genere un objeto que guarde en una lista todos los archivos alojados en `dump` utilizando `glob.glob` y `os.getcwd()` para extraer las rutas absolutas. Posteriormente genere un objeto `pd.DataFrame` que contenga todos los `csv`.

- Asegúrese de eliminar la columna `Unnamed: 0` que se genera por defecto.

Ejercicio 2: Matriz de ocurrencias

- Importe la clase `CountVectorizer` dentro de los módulos `feature_extraction.text` de la librería `sklearn`.

- Aplique la clase para extraer las 5000 palabras más repetidas en toda la base de datos.

- Con la clase inicializada, incorpore las letras con el método `fit_transform` y guarde los resultados en un nuevo objeto.

Ejercicio 3: Entrenamiento del Modelo

- Importe `sklearn.decomposition.LatentDirichletAllocation` y `sklearn.model_selection.GridSearchCV`.

- Genere una búsqueda de grilla con los siguientes hiper parámetros:

    - `n_components: [5, 10, 15]`.
    - `learning_decay: [0.7, 0.5]`.

- Entrene la búsqueda de grilla con las letras en un formato vectorizado con `CountVectorizer`.

- Reporte brevemente cuál es la mejor combinación de hiper parámetros.

Digresión: Latent Dirichlet Allocation

Latent Dirichlet Allocatio (LDA) es un modelo probabilístico generativo basado en Inferencia Variacional EM. La principal utilidad de esto es la identificación de tópicos en un corpus de texto. El proceso de inferencia se puede resumir en los siguientes pasos:

- Cada documento dentro del corpus se puede entender como una mezcla de tópicos **comunes a nivel de corpus**.
- Esta mezcla de tópicos es latente: sólo observamos los documentos registrados y sus palabras.

  La API de `sklearn.decomposition.LatentDirichletAllocation` presenta la misma funcionalidad de todo modelo de sklearn. Algunos puntos a considerar en la inicialización de la clase son:

    - `n_components`: Cantidad de tópicos a inferir en un corpus.

    - `learning_method`: Forma en la que entran los datos en entrenamiento. Cuando es 'batch', se ingresa la matriz de entrenamiento completa. Cuando es 'online', la matriz de entrenamiento ingresa de manera secuencial en parcelas pequeñas.

    - `learning_decay`: Tasa de aprendizaje en la función de pérdida. Cuando se implementa con `learning_method='online'`, el modelo se entrena con Gradiente Estocástico Descendente.

- `Perplejidad`: Busca aproximar el número óptimo de tópicos a inferir. Técnicamente evalúa qué tan bien predice una muestra específica. En función a un número de tópicos, define la distribución teórica de palabras representada por los tópicos y la compara con la ocurrencia empírica de palabras en tópicos.

Ejercicio 4 : Inferencia e Identificación de Tópicos

- En base a la mejor combinación de hiper parámetros, entrene el modelo con la matriz de atributos de las letras.

- Para identificar de qué se trata cada tópico, necesitamos identificar las principales 15 palabras asociadas con éste. Puede implementar la siguiente línea de código para identificar las principales palabras en un tópico:

```
# mediante .components_ podemos extraer una matriz que entrega las
distribución de palabras por cada tópico.
for topic_id, topic_name in enumerate(fit_best_lda.components_):
   # para cada tópico
   print("tópico: {}".format(topic_id + 1))
   # mediante argsort logramos ordenar los elementos por magnitud
   # para los elementos más relevantes ordenados por argsort, buscamos
su correlativo
   # en la matriz dispersa y devolvemos el nombre.
   # finalmente concatenamos las palabras
   print(" ".join([counter.get_feature_names()[i] for i in
topic_name.argsort()[:-15 - 1: -1]]))
```

- Comente a qué tópicos está asociada cada clase inferida.

Ejercicio 5: Identificación de probabilidades

- En base a la información generada, es posible identificar cuales van a ser los géneros más probables de ocurrir para un artista.

- Para ello necesitamos guardar la probabilidad de cada canción en nuestra base de datos original. Podemos implementar esto de la siguiente manera:

```
# generamos una transformación de los datos a distribución de tópico por
palabra en el documento
fit_best_lda = best_lda.transform(transformed_feats)

# esta transformación la podemos coaccionar a un dataframe de la
siguiente manera

topics_for_each_doc = pd.DataFrame(
   # pasamos esta matriz y la redondeamos en 3 decimales
   np.round(fit_best_lda, 3),
   # agregamos un índice
   index=df_lyrics.index
)

#agregamos identificadores de columna

topics_for_each_doc.columns = list(map(lambda x: "T: {}".format(x),
range(1, best_lda.n_components + 1)))

# concatenamos las probabilidades de tópico por documento a nuestra
matriz original

concatenated_df = pd.concat([df_lyrics, topics_for_each_doc], axis=1)
# argmax en la matriz de tópicos
concatenated_df['highest_topic'] = np.argmax(topics_for_each_doc.values,
axis=1) + 1
```

- Genere una matriz de correlaciones entre la probabilidad de tópicos inferidos. Comente brevemente cuales son las principales asociaciones existentes.

- Con esta nueva base de datos, identifique las probabilidades de pertenencia para un artista específico.

- Grafique la distribución de las probabilidades para algún artista en específico.