CLASIFICACIÓN DE TEXTOS DE NOTICIAS DE DIARIOS NACIONALES EN BASE A SU CONTENIDO

Por: Bernick Lincoln Salvador Rosas, Jhon Vargas Reyes, Miguel Ruiz Adarmes

Maestría de Ciencia de la Computación - Curso: Procesamiento de Lenguaje Natural

Facultad de Ciencias, Universidad Nacional de Ingeniería, Lima, Perú

Lima, 22 de Agosto de 2021

RESUMEN

La intención de este trabajo es la de identificar el grupo de clasificación de textos de noticias de medios nacionales mediante la predicción del grupo de pertenencia usando para ello un modelo de clasificación supervisado previamente entrenado y comparar los resultados obtenidos con la aplicación de un modelo de clusterización LDA.

Palabras clave: Procesamiento de lenguaje natural, Aprendizaje profundo, aprendizaje automático, LDA.

DEFINICIÓN DEL PROBLEMA

El problema de identificar el tópico de un texto de noticia (Topic Modeling) puede llegar a ser un problema complejo en caso no se tiene una categoría o etiqueta conocida previamente con los cuales se puede implementar un modelo de clasificación (supervisado). Sin embargo, en los últimos años se han desarrollado nuevos métodos no supervisados que permiten encontrar los tópicos de los textos a pesar de no conocerse sus categorías previas (LDA). Por ello, surge la necesidad de conocer si realmente este último tipo de modelos describen adecuadamente las categorías de los grupos de textos de noticias a las cuales pertenecen.

RECOLECCIÓN Y LIMPIEZA DE TEXTOS

Para realizar el trabajo, primero se definió, mediante palabras clave, los estilos que consideramos interesantes de acuerdo a percepciones subjetivas de tipos de noticias que se encuentran en los medios nacionales publicados en internet, estas palabras claves identificadas serán consideradas como categorías o grupos de redacción.

Las categorías o grupos de redacción seleccionados son:

- Deportes.
- Economía
- Política.
- Salud.
- Tecnología

Estas categorías fueron buscados en los siguientes medios nacionales:

- El Comercio
- Trome
- Perú 21

Para cada uno de los sitios web de cada uno de estos medios nacionales, se identificó el link que los agrupa, en cada uno de estos links se pudo apreciar que solo se mostraban un subgrupo de noticias particionando la información en diversas sub páginas identificadas por un número entero secuencial que iniciaba en 1 y terminaba en valor mayor que 1 de acuerdo al universo de noticias relativas de cada medio. Entonces conociendo este detalle, se recuperó, por cada estilo y para cada medio noticioso, el universo de noticias presente en cada uno de sus sub links.

Por lo que para cada link identificado se pudo identificar el universo de páginas por cada uno de las categorías de noticias:

- https://elcomercio.pe/noticias/deporte/ hay 9 paginas de noticias
- https://elcomercio.pe/noticias/economia/ hay 17 paginas de noticias
- https://elcomercio.pe/noticias/politica/ hay 7 paginas de noticias
- https://elcomercio.pe/noticias/salud/ hay 92 paginas de noticias
- https://elcomercio.pe/noticias/tecnologia/ hay 10 paginas de noticias
- https://trome.pe/noticias/deporte/ hay 5 paginas de noticias
- https://trome.pe/noticias/economia/ hay 5 paginas de noticias
- https://trome.pe/noticias/politica/ hay 1 paginas de noticias
- https://trome.pe/noticias/salud/ hay 39 paginas de noticias
- https://trome.pe/noticias/tecnologia/ hay 8 paginas de noticias

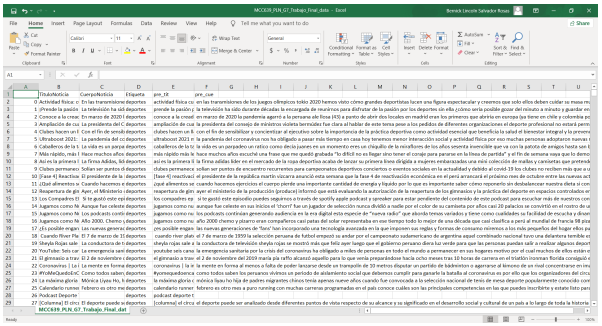
- https://peru21.pe/noticias/deporte/ hay 4 paginas de noticias
- https://peru21.pe/noticias/economia/ hay 44 paginas de noticias
- https://peru21.pe/noticias/politica/ hay 26 paginas de noticias
- https://peru21.pe/noticias/salud/ hay 25 paginas de noticias
- https://peru21.pe/noticias/tecnologia/ hay 17 paginas de noticias

Se debe notar que la categoría ya estaba identificada, por lo que en el proceso a la vez que se recuperaba cada link de noticias también se la asociaba con su categoría. El número total de links de noticias recuperadas fue de 14051, luego mediante un proceso de depuración se identificaron aquellas noticias que no contenían texto (videos o audios) que fueron eliminados dando como resultado 11208 noticias.

Luego, con la información obtenida se procedió a recuperar los títulos y los textos de cada una de las noticias usando la técnica de scraping. Este proceso duró cerca de 4 horas en un computador con procesador intel I7 con 8 Mb de RAM.

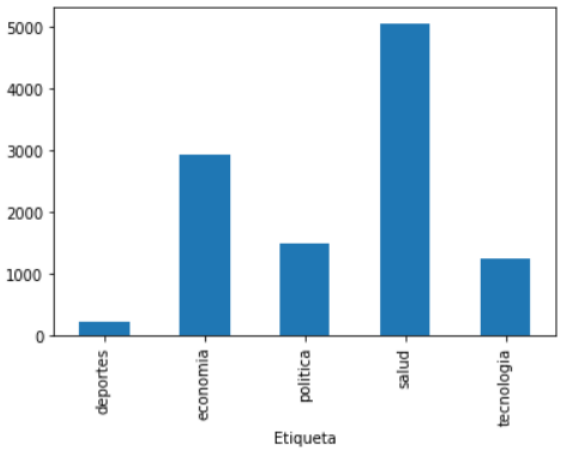
Seguidamente se realizó un pre procesamiento de limpieza básico de la data recuperada convirtiendo todo el texto a minúscula y eliminando carácteres no útiles para procesos posteriores (`.,:;!¡`).

Con la finalidad de optimizar el tiempo de los siguientes procesos, se guardó la data recuperada en un archivo externo con formato csv.



Luego de la carga de datos desde el archivo csv, se requirió eliminar los elementos de contenido nulo y los duplicados. La existencia de elementos duplicados puede ser explicado debido a que una misma noticia puede ser categorizada en diferentes categorías. Luego de llevar a cabo esto, se obtuvieron 10931 textos de noticias.

Finalmente, la distribución del número de noticias por categoría quedó de la forma: 5067 (salud), 2932 (economía), 1488 (política), 1228 (tecnología) y 216 (deportes).



Como información adicional se realizó el cálculo de frecuencias y nube de palabras, identificándose el número de veces que aparecen las palabras más populares:

- 'si', 11721
- 'persona', 10145
- 'puede', 9310
- 'salud', 9217
- 'ser', 8163
- 'años', 7969
- 'año', 6035
- 'solo', 5882
- 'perú', 5670
- 'según', 5428

PRE PROCESAMIENTO DE LOS TEXTOS

Para la parte de pre-procesamiento, se generó un dataset que tenga cómo componentes a TituloNoticia, CuerpoNoticia, Etiqueta:

	TituloNoticia	CuerpoNoticia	Etiqueta
8305	Pasos Perdidos	No solo la ciudadanía ha quedado más confund...	politica
8360	Más información en Política	Humala saluda a nuevo presidente de Paraguay...	politica
1153	Cualquiera es un señor, cualquiera ladrón, por...	Tal como van las cosas con las investigacione...	politica
5791	Coctelera.21	Nadal y Federer van por la coronaRafael Nada...	deportes
1155	Los artistas y la política, por Arturo Maldonado	El domingo fue la gala de los Globos de Oro, ...	politica

Luego se juntaron las columnas TituloNoticia y CuerpoNoticia para generar la columna prepNoticia:

```
[66] df.sample(5)
```

	Etiqueta	prepNoticias	prepNoticias_token
9982	salud	tratamiento para dejar de fumar la adicción a...	tratamiento dejar fumar adicción tabaco alcoho...
5926	economia	con la vacuna habrá un gran rebote en economía...	vacuna gran rebote economía peruana señala enz...
6672	economia	tips para emprendedores ventasdel 25 al 27 de...	tip emprendedores ventasdel marzo cámara comer...
6717	economia	barack obama "se perderán 750000 empleos por l...	barack obama perderán empleos estúpidos recort...
4556	salud	¿por qué crees que tu teléfono vibra cuando no...	cree teléfono vibra hace seguir tecnocienciae...

luego se utilizaron las siguientes técnicas tales cómo:

1. Limpieza de caracteres, stopwords, símbolos, espacios y signos de puntuación, Borrado de columnas null o vacías
2. Lemmatization, Tokenization
3. Balanceo de datos
4. One-Hot Encoding
5. Nube de palabras

A continuación describiremos los más importantes:

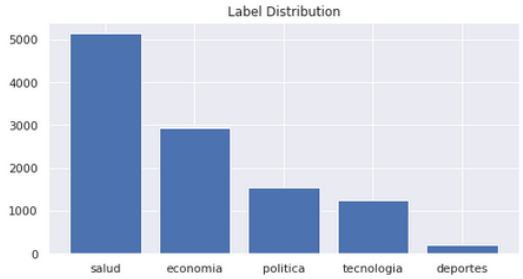
1. Balanceo de Datos:

Se encontró que las etiquetas estaban desbalanceadas. Esto puede generar sesgos a la hora del entrenamiento y posiblemente no generalize muy bien el modelo.

En la siguiente gráfica se muestran los datos no balanceados:

```
[69] val_count = df.Etiqueta.value_counts()

     plt.figure(figsize=(8,4))
     plt.bar(val_count.index, val_count.values)
     plt.title("Label Distribution")
```

```
Text(0.5, 1.0, 'Label Distribution')
```

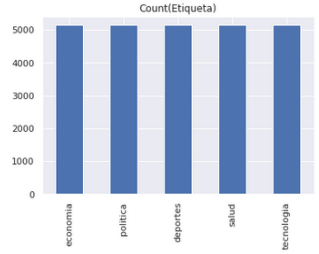


Para balancear los datos se utilizó la técnica de "over-sampling" en la cual aplicamos para que las otras etiquetas tengan el mismo tamaño que la etiqueta de mayor valor, que es la etiqueta de salud.

En la siguiente gráfica se muestran los datos balanceados:

```
print('Random over-sampling:')
print(df_balanced.Etiqueta.value_counts())

df_balanced.Etiqueta.value_counts().plot(kind='bar', title='Count(Etiqueta)');
```

```
Random over-sampling:
economia      5141
politica      5141
deportes      5141
salud         5141
tecnologia    5141
Name: Etiqueta, dtype: int64
```



2. Lematización y Tokenización

En está etapa generamos los lemas y tokens del corpus para poder medir el tamaño del vocabulario y del corpus en sí mismo.

Cabe mencionar que previo a este paso se realizó la limpieza de stopwords, signos de puntuación, espacios libres tanto en columnas y filas:

```
# Preprocesar
def text_prep_Espanol(text:str):

  # para normalizar el texto a unicode utilice lo siguiente
  text_prep= unicodedata.normalize("NFKD", text)


  # Se transforma a todas las palabras en minúscula
  text_prep = text.lower()
  # Se eliminan los caracteres '.', ',', ':' y ';'
  text_prep = re.sub('[.,:;]', '', text_prep)
  # Se eliminan los caracteres '!' y '¡'
  text_prep = re.sub('[!¡]', '', text_prep)

  return text_prep
```

En la siguiente figura podemos ver el detalle del corpus:

```
#Cargamos el Tokenizer
tokenizer = Tokenizer()
#mapear todas las palabras
tokenizer.fit_on_texts(df.prepNoticias_token)

word_index = tokenizer.word_index
vocab_size = len(tokenizer.word_index) + 1

# Información sobre el corpus pre-procesado
tokens = ' '.join(sentences_prep).split()
print(f"Número de tokens: {len(tokens)}")
print(f"Tamaño de vocabulario: {vocab_size}")
print("10 palabras más frecuentes: ")
cnt = Counter(tokens)
cnt.most_common(10)
```

```
Número de tokens: 2313797
Tamaño de vocabulario: 108977
10 palabras más frecuentes:
[('si', 11901),
 ('persona', 10228),
 ('puede', 9405),
 ('salud', 9116),
 ('ser', 8302),
 ('años', 8015),
 ('año', 6009),
 ('solo', 5942),
 ('perú', 5623),
 ('según', 5497)]
```

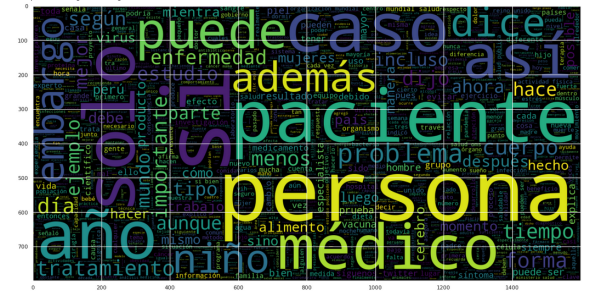
Cabe mencionar que este corpus considera la información recolectada de 5 diarios de 5 tipos de noticias: deporte, salud, tecnología, economía y política.

3. Nube de palabras por tipo de noticia

Tipo salud:

```
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".join(df[df.Etiqueta == 'salud'].prepNoticias_token))
plt.imshow(wc , interpolation = 'bilinear')
```

```
<matplotlib.image.AxesImage at 0x7fdc5a6fa1d0>
```



Tipo deporte:

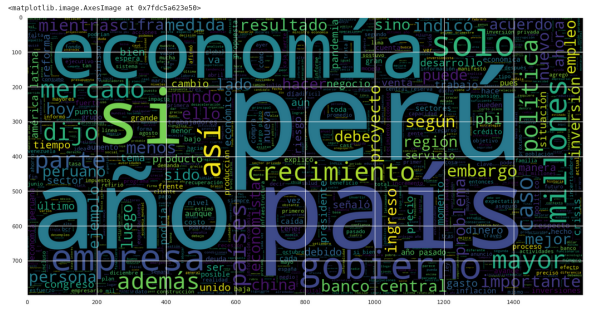
```
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".join(df[df.Etiqueta == 'deportes'].prepNoticias_token))
plt.imshow(wc , interpolation = 'bilinear')
```

```
<matplotlib.image.AxesImage at 0x7fdc5a794f50>
```

Tipo economía:

```
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".join(df[df.Etiqueta == 'economia'].prepNoticias_token))
plt.imshow(wc , interpolation = 'bilinear')
```

```
<matplotlib.image.AxesImage at 0x7fdc5a623e50>
```



4. One-hot Encoding

Un paso importante antes de entrenar el modelo es el one-hot encoding que se tiene que hacer a los features(X) y las etiquetas(y) debido a que son palabras y no números.

Para el caso del cuerpo de noticias(features X), tenemos que tokenizarlos por sentencia y luego regularizar el tamaño de cada sentencia. En este caso hemos utilizado un max-len de 50.

```
#one-hot encoding a las sentencias
sequences = tokenizer.texts_to_sequences(df.prepNoticias_token)

#sequences[1], sequences[11], df.prepNoticias_token[1], df.prepNoticias_token[11]
seq_maxlen = 50
sequences = pad_sequences(sequences, maxlen=seq_maxlen)
sequences = array(sequences)

print("X shape:", sequences.shape)
```

```
X shape: (11115, 50)
```

Para el caso de la etiquetas debemos pasarlo a un LabEncoder que luego lo utilizaremos en la capa final junto con la función de activación softmax:

```
encoder = LabelEncoder()
encoder.fit(df.Etiqueta.to_list())

y = encoder.transform(df.Etiqueta.to_list())
y = np_utils.to_categorical(y)
#y = y.reshape(-1,1)

print("y shape:", y.shape)
```

```
y shape: (11115, 5)
```

MODELO DE CLASIFICACIÓN SUPERVISADO

Para el modelo se realizó un split de 20% para generar data de entrenamiento y de validación:

```
#División entrenamiento/predicción
X_tr, X_val, y_tr, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Longitud de la secuencia de entrada
seq_length_X_tr = X_tr.shape
seq_length_X_val = X_val.shape
seq_length_y_tr = y_tr.shape
seq_length_y_val = y_val.shape

print('X Train shape:', seq_length_X_tr, '\nX Val shape:', seq_length_X_val, '\ny Train shape:', seq_length_y_tr, '\ny Val shape:', seq_length_y_val)
```

```
X Train shape: (8892, 50)
X Val shape: (2223, 50)
y Train shape: (8892, 5)
y Val shape: (2223, 5)
```

Para poder clasificar los 5 tipos de noticias, se creó el modelo Recurrente de capas LSTM y GRU para

```
model = Sequential()
model.add(Embedding(vocab_size, 100,
input_length=seq_length))
model.add(LSTM(256, return_sequences=True,
recurrent_dropout=0.1, dropout=0.1,
kernel_regularizer=regularizers.l2(0.0001),
use_bias=False))
model.add(BatchNormalization())
model.add(GRU(256,recurrent_dropout=0.1, dropout=0.1,
kernel_regularizer=regularizers.l2(0.0001),
use_bias=False))
model.add(BatchNormalization())
model.add(Dense(100, activation='relu',
kernel_regularizer=regularizers.l2(0.0001),
use_bias=False))
model.add(BatchNormalization())


model.add(Dropout(0.1))


model.add(Dense(5, activation='softmax'))
```

Cabe mencionar que antes de ingresar a la capa LSTM, se tuvo que crear una “Capa de Embedding” que la utilizamos para vectorizar las sentencias creadas anteriormente:

```
model.add(Embedding(vocab_size, 100,
input_length=seq_length))
```

En la capa de Embedding, se hicieron pruebas con vectores de 300 dimensiones, pero el resultado es el mismo que con vectores de 100 dimensiones.

El modelo tiene las siguientes características:

- En la capa de Recurrencia, se agregó una capa LSTM y GRU. También se agregó hyperparameters como dropout, kernel_regularizer(L2), recurrent_initializer, bach_normalization todo esto con el objetivo de evitar el overfiting.
- En la capa Fully Connected, se gregó 2 capas con inicializador de pesos, kernel_regularizer(L2), pero usando 1 capa Fully connected también funcionó bien.

- En la capa final, se tiene una función activación de softmax.

Layer (type)	Output Shape	Param #
embedding_21 (Embedding)	(None, 50, 100)	10897700
lstm_19 (LSTM)	(None, 50, 256)	364544
batch_normalization_37 (Batc	(None, 50, 256)	1024
gru_4 (GRU)	(None, 256)	393216
batch_normalization_38 (Batc	(None, 256)	1024
dense_50 (Dense)	(None, 100)	25600
batch_normalization_39 (Batc	(None, 100)	400
dropout_18 (Dropout)	(None, 100)	0
dense_51 (Dense)	(None, 5)	505

Total params: 11,684,013
Trainable params: 11,682,789
Non-trainable params: 1,224

None

Para hacer el entrenamiento se utilizó el optimizer "Adam" junto con 50 epocas:

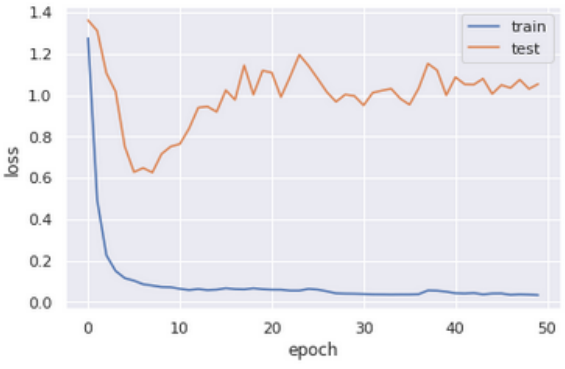
```
n_epochs = 50

model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])


history = model.fit(X_tr, y_tr, batch_size=128,
epochs=n_epochs, validation_data=(X_val, y_val))
```

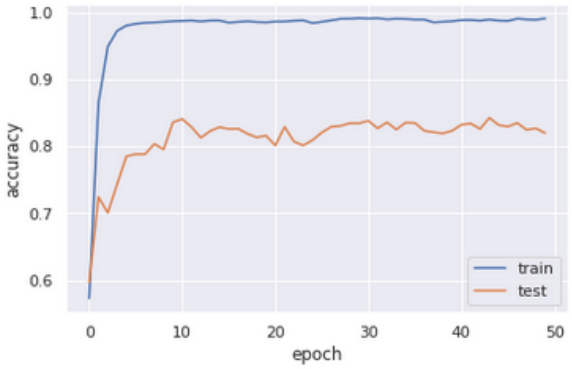
Luego del entrenamiento, se tuvo un Loss de 1.05 y un accuracy de 0.82:

70/70 [==============================] - 3s 44ms/step - loss: 1.0535 - accuracy: 0.8196

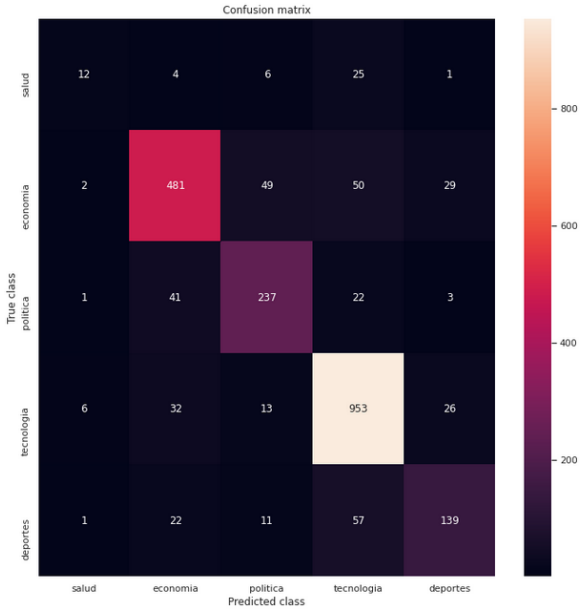
Loss = 1.05

Accuracy = 0.82





Analizando la matriz de confusión, vemos que el accurracy es de 0.82, pero aún falta mejorar algunos parámetros relacionados a las etiquetas de salud:



	precision	recall	f1-score	support
0	0.55	0.25	0.34	48
1	0.83	0.79	0.81	611
2	0.75	0.78	0.76	304
3	0.86	0.93	0.89	1030
4	0.70	0.60	0.65	230
accuracy			0.82	2223
macro avg	0.74	0.67	0.69	2223
weighted avg	0.81	0.82	0.81	2223

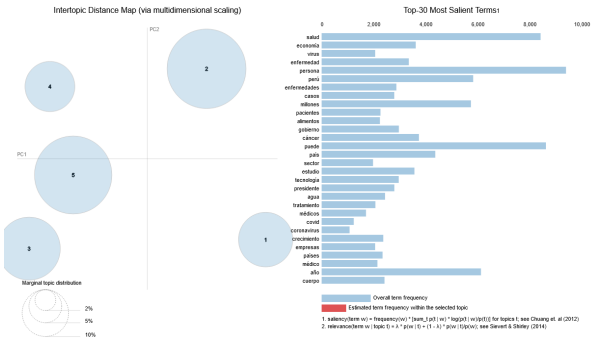
MODELO LDA

Primero se obtiene el diccionario y el corpus tipo bow a partir de los textos:

```
from gensim import corpora

dictionary = corpora.Dictionary(df['prepNoticias3'])
corpus = [dictionary.doc2bow(text) for text in df['prepNoticias3']]
```

A partir de esto, se implementa un modelo LDA con 5 tópicos basado en el conocimiento previo de las categorías de noticias.

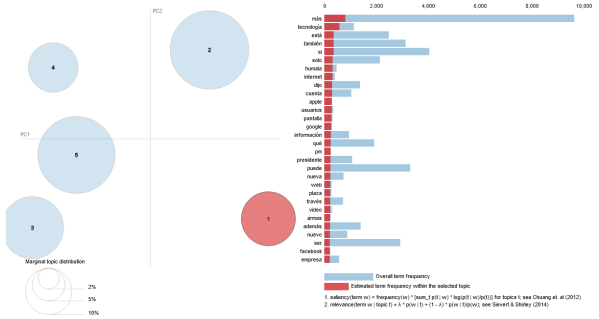
Se obtiene un Coherence Score de 0.3968 y una clusterización de dos dimensiones como se muestra en la siguiente figura:



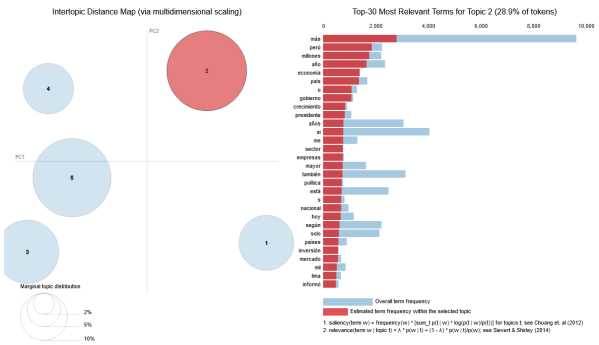
Aparentemente, la agrupación de los textos puede ser explicado adecuadamente con los 5 tópicos identificados ya que no se perciben elementos en común entre ellos.

Analizando las palabras contenidas en cada tópico:

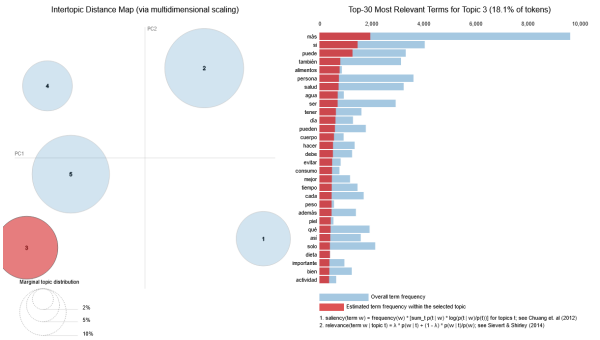
1. Las palabras más relevantes contenidas en el tópico 1 son: ‘tecnología’, ‘internet’, ‘pantalla’, ‘google’, etc. En base a estas palabras se puede asumir que este tópico corresponde a la categoría “Tecnología”.



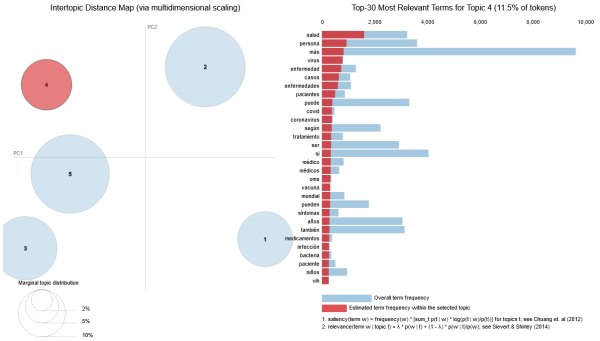
2. Las palabras más relevantes contenidas en el tópico 2 son: ‘perú’, ‘millones’, ‘economía’, ‘país’, ‘gobierno’, ‘crecimiento’, ‘empresas’, etc. En base a estas palabras se puede asumir que este tópico corresponde a las categorías “Economía” o “Política”.



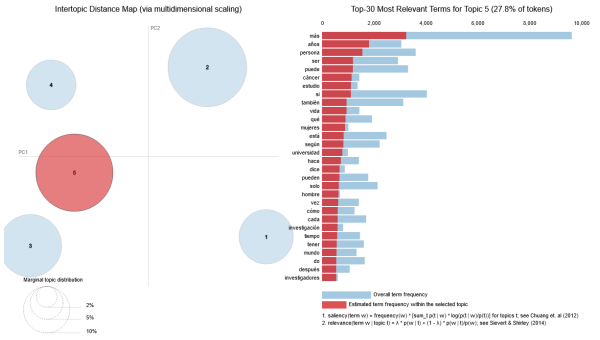
3. Las palabras más relevantes contenidas en el tópico 3 son: ‘alimentos’, ‘persona’, ‘salud’, ‘agua’, etc. En base a estas palabras se puede asumir que este tópico corresponde a las categorías “Deportes” o “Salud”.



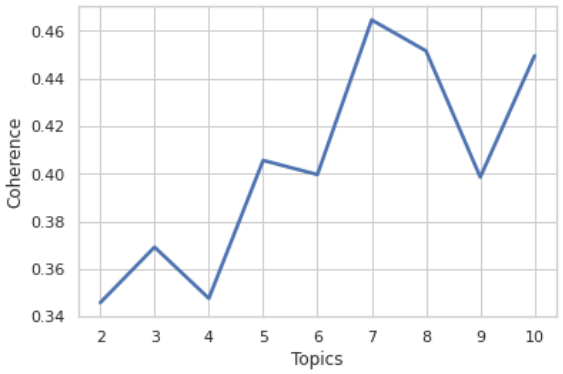
4. Las palabras más relevantes contenidas en el tópico 4 son: ‘salud’, ‘persona’, ‘virus’, ‘enfermedad’, ‘casos’, ‘pacientes’, etc. En base a estas palabras se puede asumir que este tópico corresponde a la categoría “Salud”.



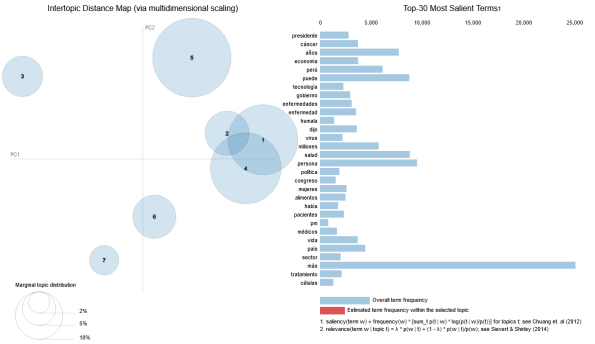
5. Las palabras más relevantes contenidas en el tópico 5 son: ‘años’, ‘persona’, ‘cáncer’, ‘vida’, ‘mujeres’, ‘universidad’, etc. En base a estas palabras no se tiene claro que este tópico corresponde a las categorías dadas.



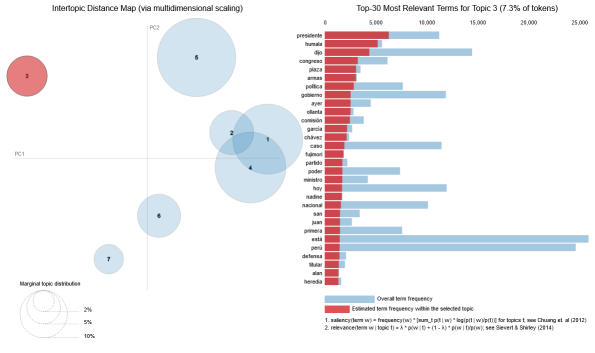
Debido a que hay tópicos identificados por el LDA que no necesariamente se asocian con las categorías de las fuentes de noticias, se procedió a obtener el número de tópicos del LDA más óptimo o el de mayor Coherence Score, para ello se implementaron diferentes modelos LDA con un número de tópicos que va desde 2 hasta 10, obteniéndose los siguientes resultados:

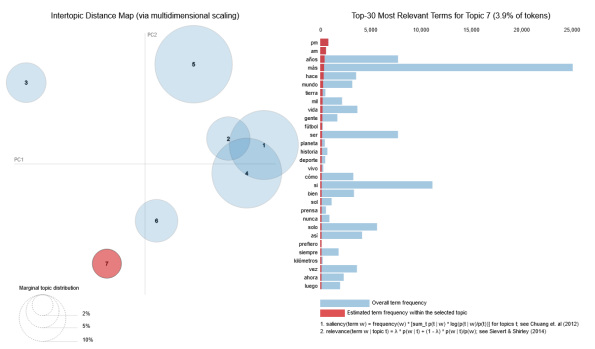


En base a estos resultados, el mejor modelo LDA (Coherence Score = 0.4646) es el de 7 tópicos



Para este modelo, las palabras contenidas en los tópicos 1, 4 y 6 se asocian bastante a la categoría "Salud", y de la misma forma los tópicos y categorías restantes: el tópico 2 a la categoría "Tecnología", el tópico 3 a la categoría "Política", el tópico 5 a la categoría "Economía" y el tópico 7 a la categoría "Deportes".





CONCLUSIONES Y LECCIONES APRENDIDAS

Sobre conclusiones para la recolección y limpieza de textos:

1. El proceso para recolección de la información demoró un poco más de 4 horas para completarse, se realizó más de una vez debido a que se perdió la conexión con google colab.
2. En cada medio nacional analizado se detectó que la mayor cantidad de páginas de noticias corresponde al grupo de salud (92 para El Comercio, 39 para el Trome). Sin embargo para Perú 21 el grupo salud ocupó el tercer lugar (25 páginas de noticias) siendo el primer lugar para economía (44 páginas de noticias) y el segundo para política (26 páginas de noticias)
3. Obviando los grupos con mayor número de noticias, se identificó que para El Comercio es más importante el grupo de economía, para Trome es deportes y economía y para Perú 21 es la política.
4. Al momento de crear modelos de clasificación de noticias usando una gran cantidad de información, vemos que genera mucha data y esto hace que se consuman mucha memoria RAM(>12G)haciendo tedioso su procesamiento y ejecución.
5. Se concluye que debido a que aún hay un loss elevado, aún falta mejorar(Tunear) el modelo.
6. El modelo LDA con 7 tópicos es el más óptimo que agrupa los textos de noticias (Coherence Score = 0.4646).
7. Los tópicos identificados con el modelo LDA en su mayoría se asocian muy bien a las categorías reales de los textos de noticias, siempre y cuando estos no contengan información difusa o que cubran más de 2 categorías. Por tanto se evidencia que los modelos LDA son sensibles a este tipo de información.

Sobre lecciones aprendidas para la recolección y limpieza de textos:

1. Para mejorar los tiempos del proceso de scraping se requiere un computador con memoria RAM superior a 8 Mb y un ancho de banda superior a 80 gbps.
2. Para mejorar el modelo se requiere analizar más medios nacionales.
3. Se tuvo que balancear los datos generados por qué generan sesgos y no generalizan correctamente el modelo luego del entrenamiento.
4. Al parecer el desbalance de los datos también afecta el resultado obtenido en la implementación del modelo LDA.
5. La redacción de noticias por categorías no necesariamente es diferente por cada medio, lo que sí afecta a la asociación entre el tópico identificado por el modelo LDA y la categoría real parece deberse a que las noticias pueden estar categorizadas en más de una 1 categoría.