# Identificación de Textos de acuerdo a su área científica.

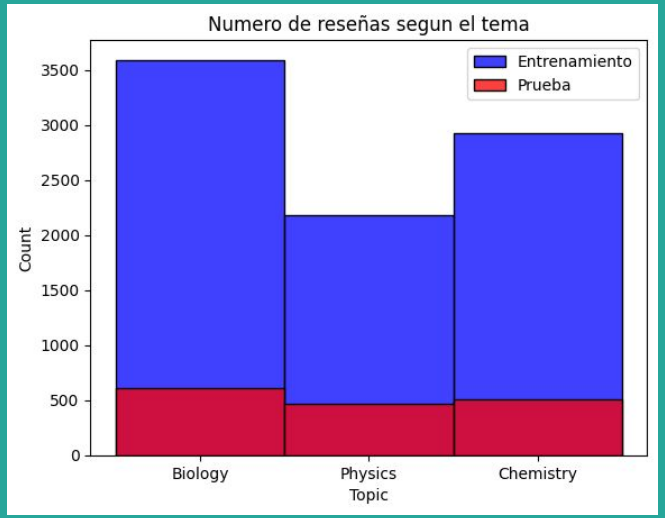
—

Alejandra Irene Hinostroza Caldas
Andrés Adrián Vargas Sánchez

# Muestreo de Datos

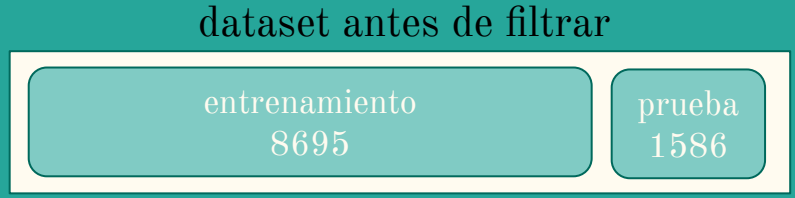
Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology

dataset antes de filtrar

# Muestreo de Datos

Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology




Se filtraron y modificaron los comentarios:

- Minúsculas

```
df_train_filt['Comment'] = df_train['Comment'].str.lower()
```

# Muestreo de Datos

Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology

dataset antes de filtrar


Se filtraron y modificaron los comentarios:

- Minúsculas
- \n, Caracteres, signos de puntuación, “Removed”, “Deleted”, Palabras vacías.

```
df_train_filt['Comment'].apply(lambda x: x.replace('\\n', ' '))
df_train_filt['Comment'].apply(lambda x: re.sub(r'[^\w\s]', ' ',x).strip())
df_train_filt[~df_train_filt['Comment'].isin(['removed', 'deleted'])]
remove stopwords from nltk package
```

# Muestreo de Datos

Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology

dataset antes de filtrar


Se filtraron y modificaron los comentarios:

- Minúsculas
- \n, Caracteres, signos de puntuación, “Removed”, “Deleted”, Palabras vacías.
- Enlaces

```
def split_link(x):
        string = re.sub(r'https?://(www\.)?', '', x) # elimina 'https://www.' del string
        string = re.sub(r'[/.-/#:=%,&\~]', ' ', string)
        return string
df_train_filt['Comment'].apply(split_link)
```

# Muestreo de Datos

Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology

dataset antes de filtrar


Se filtraron y modificaron los comentarios:

- Minúsculas
- \n, Caracteres, signos de puntuación, “Removed”, “Deleted”, Palabras vacías.
- Enlaces
- “Comentarios vacios”

```
Then no
How’s he doing now
It's not.
Why?
No
```

```
You do this 😬👌
$$$
No it’s not
😅😂😂😂
d
```

# Muestreo de Datos

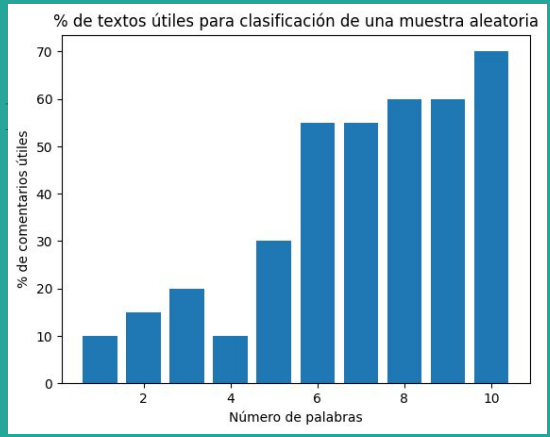
Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology

dataset antes de filtrar


Se filtraron y modificaron los comentarios:

- Minúsculas
- \n, Caracteres, signos de puntuación, “Removed”, “Deleted”,
- Enlaces
- “Comentarios vacios”
- Comentarios con pocas palabras

```
df_train_filt[df_train_filt['word_count']>=6]
```



# Muestreo de Datos

Total de Muestras: 10281
3 clases: Physics, Chemistry y Biology

dataset antes de filtrar


Se filtraron y modificaron los comentarios:

- Minúsculas
- \n, Caracteres, signos de puntuación, “Removed”, “Deleted”, Palabras vacías.
- Enlaces
- “Comentarios vacios”
- Comentarios con pocas palabras
- Tokenización y estandarización

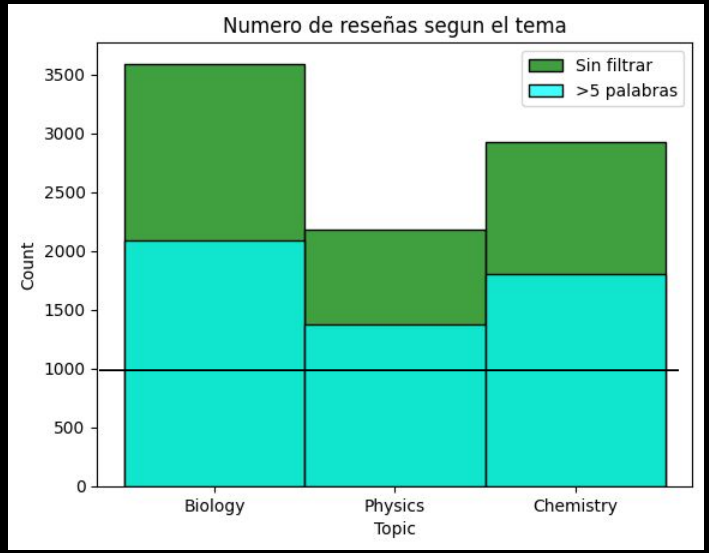
```
# categorias en one hot encoding
1 0 0 : Biology, 0 1 0 : Chemistry, 0 0 1 : Physics
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df_train_filt.Comment.tolist())
word_index = tokenizer.word_index
      -> {1: 'would', 2: 'like', 3: 'one', 4: 'get', …}
tokenizer.texts_to_sequences(df_train_filt.Comment)
pad_sequences(_test_sequence, maxlen=max_sequence_length)
```

# Dataset

Tras preprocesar los datos se dividió el dataset en:

- Para el conjunto de entrenamiento se eligieron aleatoriamente 1000 muestras de cada categoría, en total se tuvieron 3000 muestras.
- Las 2267 muestras restantes del conjunto de entrenamiento original se destinaron al conjunto de validación.
- Test: dataset aparte con 1586 muestras.




```
# tanto en el conjunto de entrenamiento como en el
# de validación se presenta un desbalance en el
# número de muestras, entonces no resulta conveniente
# usar accuracy como métrica, eligimos precision en
# su lugar
```

# Hiperparametros

- Algunos hiperparámetros del modelo se eligieron mediante un GridSearch empleando para este fin tensorboard.

```
HP_NUM_UNITS    = hp.HParam('n_units', hp.Discrete(['64,64,64','32,64,32,64']))
HP_DROPOUT      = hp.HParam('dropout', hp.Discrete([0.0,0.05,0.1,0.25]))
HP_LEARNINGRATE = hp.HParam('learning_rate', hp.Discrete([0.01, 0.001]))
HP_ACTIVATION   = hp.HParam('activation', hp.Discrete(['relu', 'tanh', 'softplus']))
HP_OPTIMIZER    = hp.HParam('optimizer', hp.Discrete(['adam', 'sgd']))
```

# Hiperparametros

- Algunos hiperparámetros del modelo se eligieron mediante un GridSearch empleando para este fin tensorboard.

| | dropout | n_units | activation | learning_rate | optimizer | precision |
|---|---|---|---|---|---|---|
| **0** | 0.25 | 64 | tanh | 0.001 | adam | 1.000000 |
| **1** | 0.25 | 64 | tanh | 0.010 | adam | 0.750000 |
| **2** | 0.05 | 64,64 | softplus | 0.010 | sgd | 1.000000 |
| **3** | 0.25 | 64,64 | relu | 0.001 | sgd | 0.454545 |
| **4** | 0.00 | 64,64,64 | relu | 0.001 | sgd | 0.387874 |
| **5** | 0.05 | 64,64,64 | relu | 0.010 | sgd | 0.385589 |
| **6** | 0.10 | 64,64,64,64 | relu | 0.001 | sgd | 0.419355 |
| **7** | 0.00 | 64,64,64,64 | softplus | 0.001 | adam | 0.382609 |

# Construcción del Modelo

```
def architecture (input_shape , n_units, activation , dropout=True, p_drop=0):

  # transformamos el string n_units en una lista
  n_units_list = [ int(x)  for x in n_units.split( ",")]

  # capa de entrada
  model = Sequential()
  model.add(Input(input_shape))

  # capas ocultas
  for i, n in enumerate(n_units_list):
    model.add(Dense(n, kernel_initializer=  'random_uniform' ,
bias_initializer= 'zeros', activation=activation, name= f'hl_{i+1}'))
    if dropout ==  True:
        model.add(Dropout(p_drop))

  # capa de salida
  model.add(Dense( 3, activation= 'softmax', kernel_initializer= 'random_uniform' ,
bias_initializer= 'zeros', name='output-layer' ))

  return model
```

Se crearon 8 modelos, se comparó su performance bajo un entrenamiento de 200 épocas, y aparte se los entrenó incluyendo el callback early_stopping.

# Funciones de pérdidas y métricas utilizadas.

La métrica elegida fue la de precisión, la cual mide la proporción de ejemplos clasificados correctamente como positivos entre todos los ejemplos clasificados como positivos por el modelo. Se calcula dividiendo los verdaderos positivos entre la suma de verdaderos positivos y falsos positivos.

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$

Accuracy no fue elegido porque estamos lidiando con una base de datos desbalanceada.

# Optimizadores.

- ADAM
- SGD

Adam tiende a converger más rápido, mientras que SGD a menudo converge a soluciones más óptimas.

# Regularizadores y dropouts.

Se utilizaron 3 dropouts en la creación del Modelo los cuales fueron:

- 0.05
- 0.1
- 0.25

# Learning rates.

Se utilizaron 2 Valores de Learning rates:

- 0.01
- 0.1

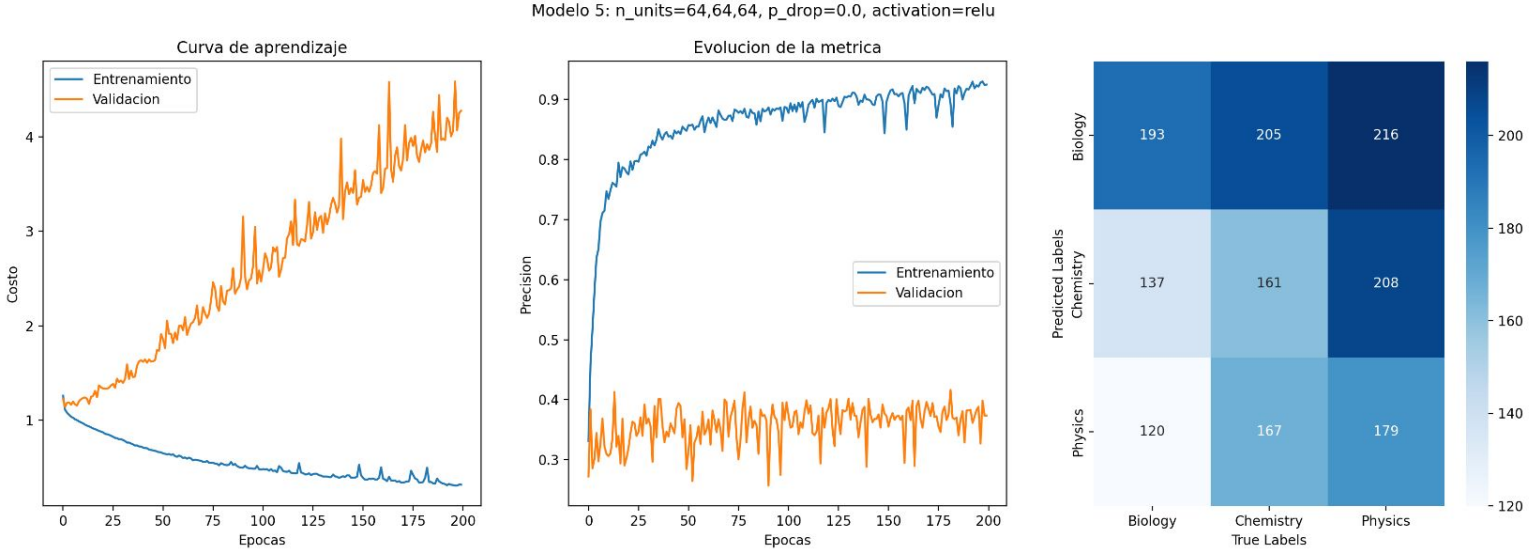
# Callbacks de Keras.

Se utilizó un el callback conocido como *early stopping:*

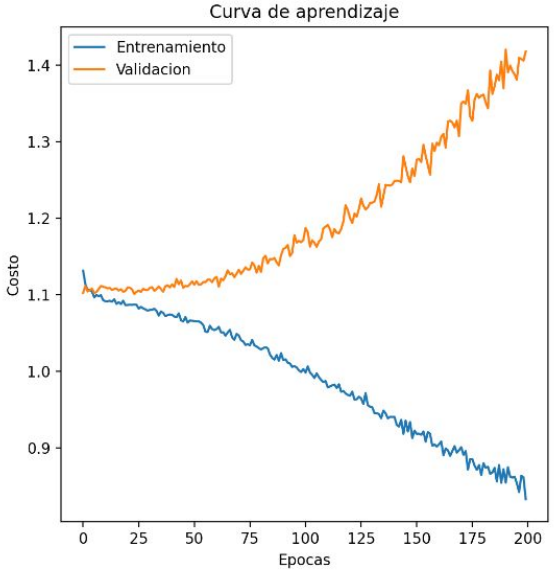
Nos sirve para evitar overfitting y ahorrar tiempo entrenando en las épocas necesarias

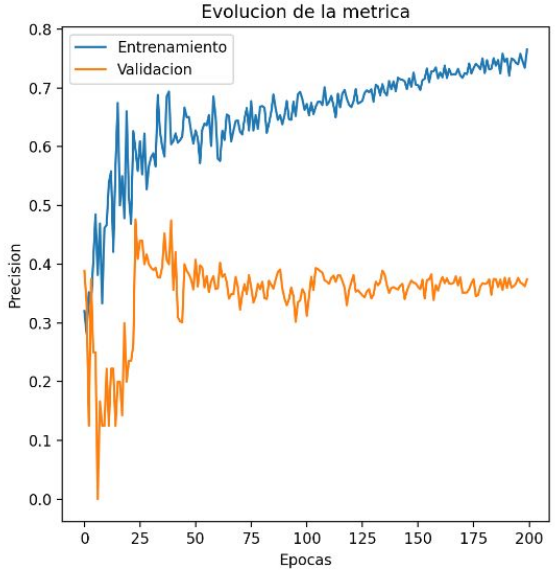
Detendrá el entrenamiento una vez que se active, pero es posible que el modelo al final del entrenamiento no sea el modelo con el mejor rendimiento en el conjunto de datos de validación.

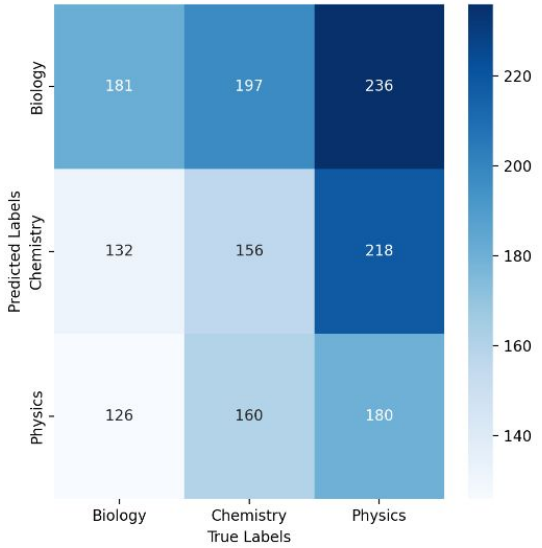
# Problemas con el sobreajuste (overfitting)



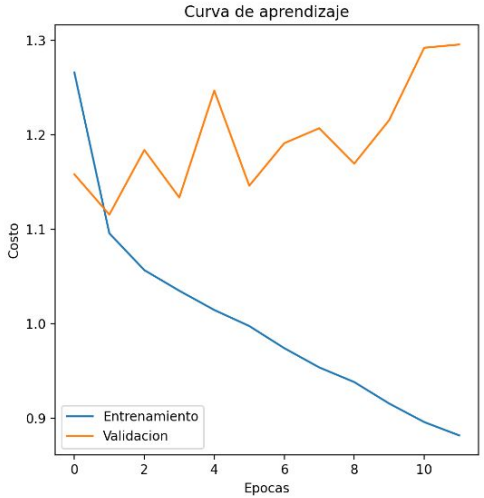
Modelo 7: n_units=64,64,64,64, p_drop=0.1, activation=relu

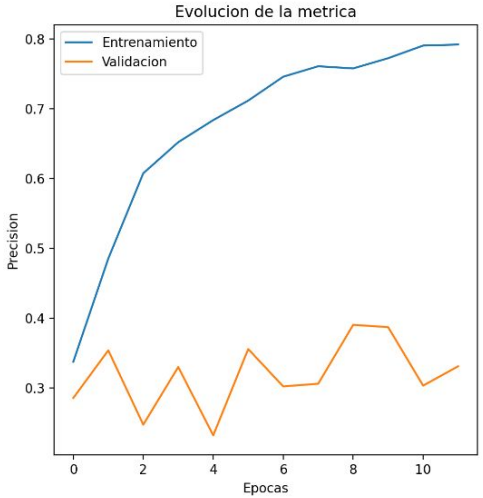


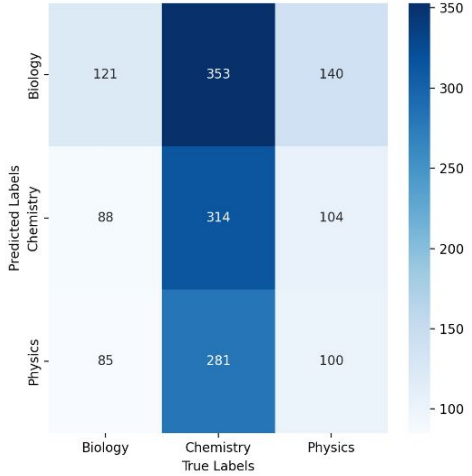



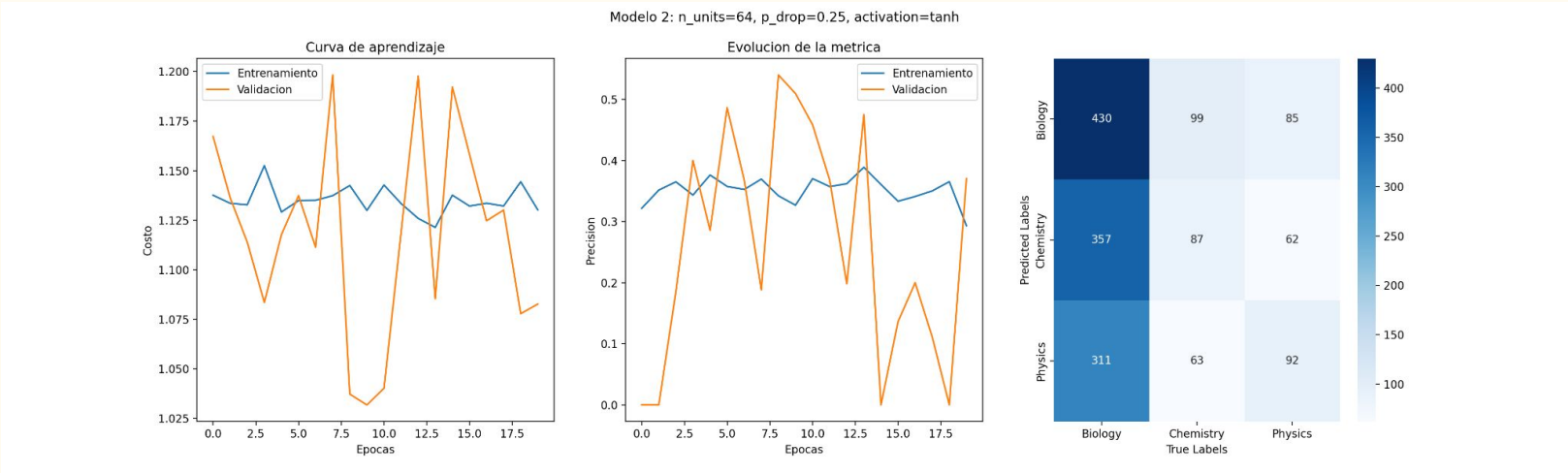
Modelo 5: n_units=64,64,64, p_drop=0.0, activation=relu



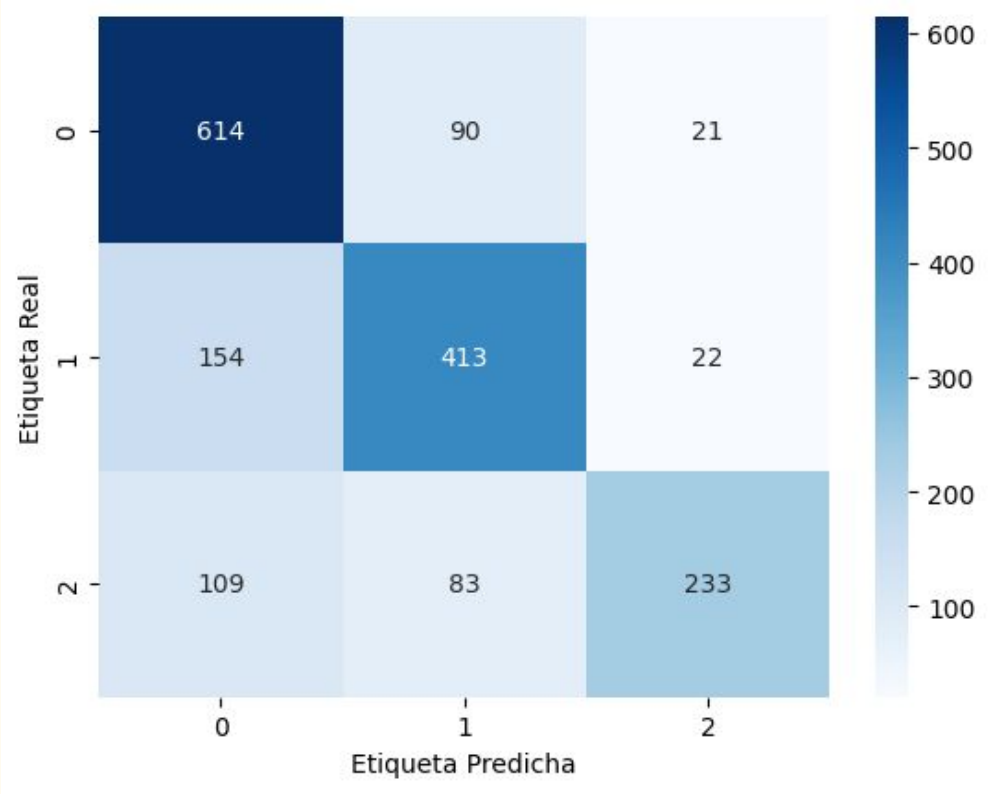



# Nuestro modelo con mejor precisión = 0.38



# SVC

Precisión del modelo: 0.7245




-Se creó un modelo SVC
-Se le dio la base de datos para entrenamiento de nuestros modelos originales y se dividió en train y test.
-Se obtuvieron mejores resultados que con nuestro mejor modelo, sin embargo la predicción no es 100% precisa.

# Conclusiones

- El filtrado redujo significativamente la información a incluir en el texto así como el tamaño del dataset
- El desbalance en los tipos de texto a clasificar se hizo evidente tanto en el dataset de entrenamiento como en el de prueba. Se optó por entrenar nuestro modelo con un subconjunto del dataset original que contuviera la misma proporción de textos de cada clase.
- no pudimos encontrar un modelo que clasifica correctamente los textos.
- Algunas soluciones para mejorar el entrenamiento:
    - Incrementar la base de datos de entrenamiento
    - Rediseñar la eliminación de palabras