TAL aplicado al análisis del discurso de los medios de prensa 📰🤓🔥

Cronograma

- Hito Unidad 2 (27 de octubre): Implementación y experimentos de varios modelos de clasificación
- Hito Proyecto (15 de diciembre): Evaluación y comparación de los modelos de los distintos equipos + integración de los mejores modelos en la arquitectura Sophia2.

índex

1. Importación del dataset
2. Balancear dataset
3. Inicialización del modelo spaCy y tokenización
4. Definición de la arquitectura CNN
5. Funciones para optimizar el modelo
6. Funciones para evaluar el modelo
7. Optimización del modelo
8. Evaluación del modelo
   8.1 Matriz de confusión


```
!spacy download es_core_news_sm
```

```
Collecting es_core_news_sm==2.2.5
  Downloading https://github.com/explosion/spacy-models/releases/download/es_core_news_s
     |████████████████████████████████| 16.2 MB 1.9 MB/s
Requirement already satisfied: spacy>=2.2.2 in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: blis<0.5.0,>=0.4.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: importlib-metadata>=0.20 in /usr/local/lib/python3.7/dist
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist
```

```
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (fron
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Building wheels for collected packages: es-core-news-sm
  Building wheel for es-core-news-sm (setup.py) ... done
  Created wheel for es-core-news-sm: filename=es_core_news_sm-2.2.5-py3-none-any.whl siz
  Stored in directory: /tmp/pip-ephem-wheel-cache-kwtxjm3j/wheels/21/8d/a9/6c1a2809c55dc
Successfully built es-core-news-sm
Installing collected packages: es-core-news-sm
Successfully installed es-core-news-sm-2.2.5
✓ Download and installation successful
You can now load the model via spacy.load('es_core_news_sm')
```

```
import time
import warnings
warnings.filterwarnings('ignore')

# Data manipulation
import re
import pandas as pd
import numpy as np
import random
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import StratifiedShuffleSplit

# Plotting
import matplotlib.pyplot as plt
from tqdm import tqdm

# NLP
import spacy
import torch
import torchtext
from torchtext import data
from torchtext import datasets
from torchtext.legacy import data

# Reports
from sklearn.metrics import confusion_matrix, classification_report

# CNN
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

1. Importación del dataset

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mou
```

2. Balancear dataset

Realizaremos random undersampling

3. Inicialización del modelo spaCy y tokenización

```
CNN_train = pd.read_csv("gdrive/My Drive/nlp-chilean-news-media/actualizado/CNN_train.csv")
CNN_valid = pd.read_csv("gdrive/My Drive/nlp-chilean-news-media/actualizado/CNN_valid.csv")
CNN_test = pd.read_csv("gdrive/My Drive/nlp-chilean-news-media/actualizado/CNN_test.csv")
```

```
spacy_es = spacy.load('es_core_news_sm')
def tokenize_es(sentence):
    return [tok.text for tok in spacy_es.tokenizer(sentence)]
```

```
TEXT = data.Field(tokenize=tokenize_es, batch_first = True)
LABEL = data.LabelField()
fields = [('content', TEXT),('label', LABEL)]

SEED = 1234

random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.backends.cudnn.deterministic = True

train_data, valid_data, test_data = data.TabularDataset.splits(
                                        path = 'gdrive/My Drive/nlp-chilean-news-media/actual
                                        train = 'CNN_train.csv',
                                        validation= 'CNN_valid.csv',
                                        test = 'CNN_test.csv',
                                        format = 'csv',
                                        fields = fields,
                                        skip_header = True
)
```

```
BATCH_SIZE = 32
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
print(device)

train_iterator, valid_iterator, test_iterator = data.BucketIterator.splits(
    (train_data, valid_data, test_data),
    batch_size = BATCH_SIZE,
    device = device,
    sort_key=lambda x:len(x.label),
    sort_within_batch=False)
```

```
cpu
```

4. Definición de la arquitectura CNN

```
!wget http://dcc.uchile.cl/~jperez/word-embeddings/glove-sbwc.i25.vec.gz
```

```
--2021-12-15 05:09:43--  http://dcc.uchile.cl/~jperez/word-embeddings/glove-sbwc.i25.vec
Resolving dcc.uchile.cl (dcc.uchile.cl)... 192.80.24.11
Connecting to dcc.uchile.cl (dcc.uchile.cl)|192.80.24.11|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.dcc.uchile.cl/~jperez/word-embeddings/glove-sbwc.i25.vec.gz [foll
--2021-12-15 05:09:43--  https://www.dcc.uchile.cl/~jperez/word-embeddings/glove-sbwc.i2
Resolving www.dcc.uchile.cl (www.dcc.uchile.cl)... 192.80.24.11, 200.9.99.213
Connecting to www.dcc.uchile.cl (www.dcc.uchile.cl)|192.80.24.11|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://users.dcc.uchile.cl/~jperez/word-embeddings/glove-sbwc.i25.vec.gz [fo
--2021-12-15 05:09:44--  https://users.dcc.uchile.cl/~jperez/word-embeddings/glove-sbwc
Resolving users.dcc.uchile.cl (users.dcc.uchile.cl)... 200.9.99.211, 192.80.24.4
Connecting to users.dcc.uchile.cl (users.dcc.uchile.cl)|200.9.99.211|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 949886421 (906M) [application/x-gzip]
Saving to: ‘glove-sbwc.i25.vec.gz.1’

glove-sbwc.i25.vec. 100%[===================>] 905.88M  9.21MB/s    in 2m 9s

2021-12-15 05:11:54 (7.01 MB/s) - ‘glove-sbwc.i25.vec.gz.1’ saved [949886421/949886421]
```

```
MAX_VOCAB_SIZE = 50000

## TENER VECTORES EN ESPAÑOL
vec = torchtext.vocab.Vectors('glove-sbwc.i25.vec.gz', cache='.')
TEXT.build_vocab(train_data, vectors=vec, max_size = MAX_VOCAB_SIZE, unk_init = torch.Tensor.

LABEL.build_vocab(train_data)
```

```
print(LABEL.vocab.stoi)
```

```
defaultdict(None, {'catástrofes y accidentes': 0, 'ciencia y tecnología': 1, 'crimen, de
```

```
class CNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, n_filters, filter_sizes, output_dim,
                 dropout, pad_idx):

        super().__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.convs = nn.ModuleList([
                                    nn.Conv1d(in_channels = 1,
                                              out_channels = n_filters,
                                              kernel_size = (fs, embedding_dim))
                                    for fs in filter_sizes
                                    ])

        self.fc = nn.Linear(len(filter_sizes) * n_filters, output_dim)
        self.dropout = nn.Dropout(dropout)

    def forward(self, text):


        embedded = self.embedding(text)
        embedded = embedded.unsqueeze(1)

        conved = [F.relu(conv(embedded)).squeeze(3) for conv in self.convs]

        pooled = [F.max_pool1d(conv, conv.shape[2]).squeeze(2) for conv in conved]

        cat = self.dropout(torch.cat(pooled, dim = 1))

        return self.fc(cat)
```

```
INPUT_DIM = len(TEXT.vocab)
EMBEDDING_DIM = 300
N_FILTERS = 100
FILTER_SIZES = [2,3,4]
OUTPUT_DIM = len(LABEL.vocab)
DROPOUT = 0.5
PAD_IDX = TEXT.vocab.stoi[TEXT.pad_token]

model = CNN(INPUT_DIM, EMBEDDING_DIM, N_FILTERS, FILTER_SIZES, OUTPUT_DIM, DROPOUT, PAD_IDX)
```

```
OUTPUT_DIM
```

```
10
```

```
def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requires_grad)
```

```
print(f'The model has {count_parameters(model):,} trainable parameters')
```

```
pretrained_embeddings = TEXT.vocab.vectors
UNK_IDX = TEXT.vocab.stoi[TEXT.unk_token]

model.embedding.weight.data.copy_(pretrained_embeddings)
model.embedding.weight.data[UNK_IDX] = torch.zeros(EMBEDDING_DIM)
model.embedding.weight.data[PAD_IDX] = torch.zeros(EMBEDDING_DIM)
```

5. Funciones para optimizar el modelo

```
optimizer = optim.Adam(model.parameters())

criterion = nn.CrossEntropyLoss() #MULTICLASS ---> en lugar de .BCEWithLogitsLoss() (Binary C

model = model.to(device)
criterion = criterion.to(device)
```

```
def train(model, iterator, optimizer, criterion, divisor):

    epoch_loss = 0
    epoch_acc = 0
    model.train()

    for batch in tqdm(iterator, desc='train'):

        optimizer.zero_grad()
        predictions = model(batch.content)

        loss = criterion(predictions, batch.label)
        acc = categorical_accuracy(predictions, batch.label, divisor)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
        epoch_acc += acc.item()

    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

```
def epoch_time(start_time, end_time):

    elapsed_time = end_time - start_time
    elapsed_mins = int(elapsed_time / 60)
    elapsed_secs = int(elapsed_time - (elapsed_mins * 60))

    return elapsed_mins, elapsed_secs
```

6. Funciones para evaluar el modelo

```
def categorical_accuracy(preds, y, divisor):
    """
    Returns accuracy per batch, i.e. if you get 8/10 right, this returns 0.8, NOT 8
    """
    max_preds = preds.argmax(dim = 1, keepdim = True) # get the index of the max probability
    correct = max_preds.squeeze(1).eq(y)

    return correct.sum() / divisor([y.shape[0]])


def evaluate(model, iterator, criterion, divisor):

    epoch_loss = 0
    epoch_acc = 0
    model.eval()

    with torch.no_grad():
        for batch in tqdm(iterator, desc='eval'):

            predictions = model(batch.content)
            loss = criterion(predictions, batch.label)
            acc = categorical_accuracy(predictions, batch.label, divisor)

            epoch_loss += loss.item()
            epoch_acc += acc.item()

    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

7. Optimización del modelo

```
print("inicio optimización")

N_EPOCHS = 3
best_valid_loss = float('inf')

for epoch in range(N_EPOCHS):

    start_time = time.time()
    divisor = torch.FloatTensor if str(device) == 'cpu' else torch.cuda.FloatTensor
    train_loss, train_acc = train(model, train_iterator, optimizer, criterion, divisor)
    valid_loss, valid_acc = evaluate(model, valid_iterator, criterion, divisor)

    end_time = time.time()
    epoch_mins, epoch_secs = epoch_time(start_time, end_time)
```

```
    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        name = 'gdrive/My Drive/nlp-chilean-news-media/actualizado/tematic-model-CNN'+'_ep'+s
        torch.save({'epoca': epoch,
                    'model_state_dict': model.state_dict(),
                    'optimizer_state_dict': optimizer.state_dict(),
                    'Valid_loss': best_valid_loss}, name)

    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    print(f'\tTrain Loss: {train_loss:.3f} | Train Acc: {train_acc*100:.2f}%')
    print(f'\t Val. Loss: {valid_loss:.3f} |  Val. Acc: {valid_acc*100:.2f}%')

best_model = CNN(INPUT_DIM, EMBEDDING_DIM, N_FILTERS, FILTER_SIZES, OUTPUT_DIM, DROPOUT, PAD_

pretrained_embeddings = TEXT.vocab.vectors
UNK_IDX = TEXT.vocab.stoi[TEXT.unk_token]

best_model.embedding.weight.data.copy_(pretrained_embeddings)
best_model.embedding.weight.data[UNK_IDX] = torch.zeros(EMBEDDING_DIM)
best_model.embedding.weight.data[PAD_IDX] = torch.zeros(EMBEDDING_DIM)

name = 'gdrive/My Drive/nlp-chilean-news-media/actualizado/tematic-model-CNN'+'_ep'+str(1)+'.
best_model.load_state_dict(torch.load(name, map_location=torch.device('cpu'))['model_state_di
```

```
inicio optimización
train: 100%|██████████| 1228/1228 [1:45:47<00:00,  5.17s/it]
eval: 100%|██████████| 267/267 [04:41<00:00,  1.06s/it]
Epoch: 01 | Epoch Time: 110m 29s
	Train Loss: 0.584 | Train Acc: 80.39%
	 Val. Loss: 0.851 |  Val. Acc: 72.57%
train: 100%|██████████| 1228/1228 [1:44:56<00:00,  5.13s/it]
eval: 100%|██████████| 267/267 [04:42<00:00,  1.06s/it]
Epoch: 02 | Epoch Time: 109m 38s
	Train Loss: 0.436 | Train Acc: 85.54%
	 Val. Loss: 0.900 |  Val. Acc: 72.67%
train:  74%|███████▍  | 912/1228 [1:20:47<29:33,  5.61s/it]
```

8. Evaluación del modelo

```
best_model = CNN(INPUT_DIM, EMBEDDING_DIM, N_FILTERS, FILTER_SIZES, OUTPUT_DIM, DROPOUT, PAD_


pretrained_embeddings = TEXT.vocab.vectors
UNK_IDX = TEXT.vocab.stoi[TEXT.unk_token]

best_model.embedding.weight.data.copy_(pretrained_embeddings)
best_model.embedding.weight.data[UNK_IDX] = torch.zeros(EMBEDDING_DIM)
best_model.embedding.weight.data[PAD_IDX] = torch.zeros(EMBEDDING_DIM)
```

```
name = 'gdrive/My Drive/nlp-chilean-news-media/actualizado/tematic-model-CNN'+'_ep'+str(3)+'.
best_model.load_state_dict(torch.load(name, map_location=torch.device('cpu'))['model_state_di
```

```
<All keys matched successfully>
```

```
prediction_test = []
labels_test=[]

for batch in tqdm(test_iterator, desc='predictions'):
    labels_test.append(batch.label.cpu().detach().numpy())
    predictions = best_model(batch.content.cpu()).squeeze(1)
    prediction_test.append(predictions.argmax(dim=1).detach().numpy())

y_true = np.concatenate(labels_test)
y_pred = np.concatenate(prediction_test)
```

```
predictions: 100%|██████████| 267/267 [04:24<00:00,  1.01it/s]
```

8.1 Matriz de confusión

```
y_true = np.concatenate(labels_test)
y_pred = np.concatenate(prediction_test)

display(y_pred,y_true)
```

```
array([9, 1, 1, ..., 2, 8, 2])
array([9, 7, 7, ..., 2, 2, 2])
```

```
cm = confusion_matrix(y_true, y_pred)
display(cm)

print(classification_report(y_true, y_pred))
```

```
array([[801,  10,  21,   2,   8,   2,   5,  21,  16,   8],
       [ 20, 411,  41,  24,  58,  24,  23,  71, 140,  33],
       [ 22,  25, 660,   5,  34,   2,  13,  27,  72,   7],
       [  4,  22,   0, 805,   4,   4,   2,   4,  12,   0],
       [ 37,  44,  48,  11, 406,  11,  87,  75,  81,  36],
       [  5,   9,   3,   4,   6, 803,  10,   2,   8,   2],
       [ 20,   9,  21,   1,  47,   7, 635,  26,  32,  15],
       [ 52,  94,  53,  13, 115,  15,  50, 264, 158,  47],
       [ 28, 107,  37,  16,  60,   6,  21,  71, 464,  26],
       [ 14,  24,   9,   5,  36,   3,  16,  58,  29, 679]])
              precision    recall  f1-score   support

           0       0.80      0.90      0.84       894
           1       0.54      0.49      0.51       845
           2       0.74      0.76      0.75       867
           3       0.91      0.94      0.92       857
           4       0.52      0.49      0.50       836
           5       0.92      0.94      0.93       852
           6       0.74      0.78      0.76       813
           7       0.43      0.31      0.36       861
           8       0.46      0.56      0.50       836
           9       0.80      0.78      0.79       873

    accuracy                           0.69      8534
   macro avg       0.68      0.69      0.69      8534
weighted avg       0.69      0.69      0.69      8534
```