Universidad Nacional de General San Martín
Escuela de Ciencia y Tecnología - Licenciatura en Ciencia de datos

EXAMEN PARCIAL — Introducción al Procesamiento de Lenguaje Natural

Fecha: _____
Apellido y nombre: _____

SECCIÓN 1. Elegir tres preguntas de esta sección y responderlas.

1. Un equipo de investigación está analizando sentimientos en reseñas de medicamentos. Aplicaron el siguiente pipeline. A continuación tiene una muestra de las primeras cinco reseñas crudas y los resultados del pipeline:

```
# Pipeline aplicado:
# 1. Tokenización por espacios
# 2. Eliminación de stopwords estándar en español
# 3. Stemming con SnowballStemmer

# Reseñas originales:
R1: '¡El tratamiento no mejoró nada! Me sentí peor que antes...'
R2: 'No recomendaría este medicamento --¿para qué?-- para nada'
R3: 'Nada que destacar: resultados mediocres, muy mediocres.'
R4: 'Me curé completamente. ¡No tengo palabras para agradecer!'
# Tokens resultantes:
R1: ['¡el', 'tratamiento', 'mejor', 'nada!', 'sent', 'peor', 'antes...']
R2: ['recomend', 'medicament', '--¿para', 'qué?--']
R3: ['nada', 'destac', 'result', 'mediocr,', 'mediocr.']
R4: ['cur', 'complet.', '¡no', 'palabr', 'agradec!']
```

a) Identifique al menos dos problemas concretos que este pipeline introduce para la tarea, usando ejemplos de los tokens resultantes. No se trata de errores de código sino de decisiones de preprocesamiento inadecuadas para esta tarea específica.
b) Un colega propone reemplazar el stemming por lematización y eliminar solo las stopwords que no sean negaciones. ¿Resuelve esto ambos problemas que identificó? ¿Hay algún problema que persista de todas formas? Justifique.
c) Dado el corpus y la tarea, ¿agregaría bigramas al pipeline? Justifique su respuesta.

2. Trabaja con el siguiente corpus de sentencias judiciales. Su objetivo es entrenar un clasificador que distinga sentencias condenatorias de absolutorias:

```
D1:  'El tribunal resolvió condenar al imputado a cinco años de prisión'
D2:  'La cámara en acuerdo con el tribunal, confirmó la condena dictada en primera
instancia'
D3:  'El tribunal absolvió al acusado por falta de pruebas suficientes'
D4:  'Se resolvió la absolución del imputado al no probarse el hecho'
D5:  'El tribunal confirmó la sentencia condenatoria de segunda instancia'
```

```
# TF-IDF calculado (fragmento, solo términos relevantes):
           condenar  condena  absolvió  absolución  tribunal  imputado
D1             0.71     0.00      0.00        0.00      0.35      0.35
D2             0.00     0.71      0.00        0.00      0.35      0.00
D3             0.00     0.00      0.71        0.00      0.35      0.00
D4             0.00     0.00      0.00        0.71      0.00      0.35
D5             0.00     0.35      0.00        0.00      0.35      0.00
```

a) Observe que 'condenar' y 'condena' aparecen como columnas separadas, igual que 'absolvió' y 'absolución'. ¿Qué problema representa esto para el clasificador? ¿Qué paso de preprocesamiento lo resolvería y por qué?
b) ¿El término 'tribunal' es un buen feature para distinguir sentencias condenatorias de absolutorias? Razone a partir de su distribución en la matriz y del concepto de IDF.

3. Una socióloga entrenó dos modelos LDA sobre 8.000 notas periodísticas argentinas (2015-2023) y obtuvo los siguientes resultados:

```
Modelo A  (k=6, coherencia c_v = 0.58):
  T0: economía, dólar, inflación, precio, mercado, reservas
  T1: gobierno, presidente, ministro, decreto, medida, política
  T2: elección, candidato, voto, campaña, partido, resultado
  T3: juicio, causa, tribunal, fiscal, condena, imputado
  T4: salud, hospital, vacuna, pandemia, caso, contagio
  T5: economía, gobierno, medida, inflación, crisis, ajuste

Modelo B  (k=10, coherencia c_v = 0.63):
  T0: dólar, reservas, brecha, tipo, cambio, devaluación
  T1: inflación, precio, canasta, índice, aumento, IPC
  T2: presidente, decreto, medida, gobierno, anuncio, cadena
  T3: congreso, diputado, senado, sesión, ley, proyecto
  T4: elección, candidato, voto, campaña, partido, primaria, resultado
  T5: juicio, causa, tribunal, fiscal, condena, imputado
  T6: salud, hospital, cama, UCI, pandemia, contagio
  T7: vacuna, dosis, inoculación, esquema, efectividad, Sputnik
  T8: economía, gobierno, crisis, ajuste, FMI, acuerdo
  T9: educación, escuela, docente, paritaria, clase, protocolo

# Salida detallada del Modelo B -- T4 (electoral):
0.052*'partido' + 0.041*'elección' + 0.035*'candidato' + 0.028*'voto' +
0.019*'campaña'
```

a) El Modelo A tiene un tópico problemático. Identifíquelo, explique exactamente qué lo hace problemático y qué lo pudo haber causado.
b) El Modelo B tiene mayor coherencia c_v y más tópicos, pero la socióloga duda si elegirlo. Formule al menos un argumento a favor del Modelo A y uno a favor del Modelo B, considerando el objetivo de investigación y no solo la métrica.
c) En la salida detallada del T4 del Modelo B, el término 'partido' tiene el mayor peso (0.052). Sin embargo, en el corpus también hay artículos deportivos donde 'partido' es frecuente. ¿Cómo maneja LDA esta ambigüedad? ¿Es un problema estructural o una limitación resoluble con más datos?
d) El Modelo B fue entrenado con `passes=2`. ¿Qué consecuencia tiene esto sobre la calidad de los tópicos?

4. Un modelo word2vec fue entrenado sobre un corpus de artículos de economía argentina. Se muestran algunas relaciones geométricas entre vectores obtenidas del modelo:

```
# Similitudes coseno entre pares de términos:
similitud('dólar', 'peso')         = 0.91
similitud('dólar', 'inflación')    = 0.87
similitud('dólar', 'fútbol')       = 0.12

# Operaciones aritméticas sobre vectores:
vec('rey') - vec('hombre') + vec('mujer')  ≈  vec('reina')   # ejemplo clásico

# Resultado en el corpus económico:
vec('exportación') - vec('producto') + vec('servicio')  ≈  vec('turismo')

# Vecinos más cercanos de 'ajuste':
['recesión', 'tarifazo', 'devaluación', 'motosierra', 'déficit', 'Milei']
```

a) La similitud entre 'dólar' y 'peso' es 0.91, mucho mayor que entre 'dólar' y 'fútbol' (0.12). ¿Qué propiedad del entrenamiento de word2vec explica este resultado? ¿Qué información del corpus captura esa similitud?
b) El modelo produce la siguiente analogía: `vec('exportación') - vec('producto') + vec('servicio') ≈ vec('turismo')`. Un investigador quiere aplicar la misma lógica para explorar relaciones de género en el corpus económico y calcula `vec('empresario') - vec('hombre') + vec('mujer')`. Obtiene como resultado más cercano `vec('secretaria')`. ¿Qué nos dice este resultado sobre el corpus y sobre los sesgos que puede capturar word2vec? ¿Qué respuesta podría ser considerada como menos sesgada?
c) El mismo investigador quiere usar estos embeddings como features para clasificar noticias en 'económicas' vs. 'no económicas'. Propone representar cada documento promediando los vectores de todas sus palabras. Responda las siguientes preguntas:

- Si el corpus tiene una dimensión de 5.000 documentos y un vocabulario de tamaño 10.000 y se utiliza un embedding de 250 dimensiones, ¿qué dimensionalidad tendrá la matriz de documentos “embebidos”?
- ¿Qué ventaja tiene esto respecto a TF-IDF?

SECCIÓN 2. Elegir dos preguntas de esta sección y responderlas

5. Observe el siguiente fragmento de código y responda las preguntas.

```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [
    'el gato come pescado',
    'el perro come carne',
    'el gato y el perro juegan',
]

vec = CountVectorizer()
X = vec.fit_transform(corpus)

print(vec.get_feature_names_out())
print(X.toarray())
```

a) Escriba la matriz `X.toarray()` resultante. Indique claramente qué representa cada fila y cada columna.
b) Si agregáramos el parámetro `ngram_range=(1,2)`, ¿cómo cambiaría el vocabulario? Mencione dos bigramas que aparecerían y explique qué ventaja aportan.

6. El siguiente código implementa un pipeline completo de clasificación de reseñas de películas (positivas / negativas). La variable que contiene el texto se llama “texto” y la que contiene las reseñas “valor”. Complete los huecos (_____) con el valor, función o argumento correcto.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

df = pd.read_csv('resenas.csv')

# 1. Separar features y etiquetas
X = df[_____________________________]
y = df[_____________________________]

# 2. Split train/test con semilla fija
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=_____________________________, random_state=42)

# 3. Vectorizar con TF-IDF (unigramas y bigramas, max 5000 features)
tfidf = TfidfVectorizer(ngram_range=_____________________________, max_features=5000)
X_train_vec = tfidf._____________________________(X_train)
X_test_vec  = tfidf._____________________________(X_test)

# 4. Entrenar clasificador
clf = LogisticRegression(max_iter=_____________________________)
clf._____________________________(X_train_vec, y_train)

# 5. Evaluar
y_pred = clf._____________________________(X_test_vec)
print('Accuracy:', accuracy_score(y_test, y_pred))
print(classification_report(_____________________________))
```

a) ¿Por qué es incorrecto aplicar `fit_transform()` también al conjunto de test? Explique el concepto de data leakage en este contexto.
b) ¿Qué métricas devuelve `classification_report`? Defina brevemente precision y recall, y explique cuándo priorizaría una sobre la otra.

7. El siguiente código entrena un modelo LDA sobre un corpus de artículos académicos. Complete los huecos marcados en amarillo con el valor, método o argumento correcto.

```
import gensim
from gensim import corpora, models
from gensim.models import CoherenceModel

# corpus_tokens: lista de listas de tokens ya preprocesados

# 1. Construir el diccionario
diccionario = corpora._____(corpus_tokens)

# 2. Filtrar extremos: descartar términos que aparecen
#    en menos de 3 docs o en más del 60% del corpus
diccionario.filter_extremes(no_below=_____, no_above=0.6)

# 3. Convertir cada documento a formato bag-of-words
bow_corpus = [diccionario._____(doc) for doc in corpus_tokens]

# 4. Entrenar LDA con 10 tópicos, 20 pasadas
lda = models.LdaModel(
    bow_corpus,
    num_topics=_____,
    id2word=diccionario,
    passes=_____,
    random_state=42
)

# 5. Imprimir las 6 palabras más relevantes de cada tópico
for idx, topic in lda.print_topics(_____):
    print(f'Tópico {idx}: {topic}')

# 6. Calcular coherencia c_v del modelo
coherencia_model = CoherenceModel(
    model=lda,
    texts=_____,        # tokens originales, no bow
    dictionary=diccionario,
    coherence=_____
)
score = coherencia_model._____()
print(f'Coherencia c_v: {score:.4f}')
```

a) ¿Qué diferencia hay entre el formato `bow_corpus` que construye `doc2bow` y la matriz TF-IDF que produce sklearn?
b) La métrica de coherencia `c_v` dio 0.42 con 10 tópicos y 0.61 con 6 tópicos. ¿Qué indica este resultado? ¿Elegiría el modelo con mayor o menor coherencia y por qué?