ULPGC
Universidad de
Las Palmas de
Gran Canaria

SIANI
Instituto Universitario de Sistemas Inteligentes
y Aplicaciones Numéricas en Ingeniería

Trabajo de Curso: Ciencia de Datos en Ingeniería

Sistema de Clasificación y Recomendación de Revistas Científicas mediante Técnicas Clásicas y Conexionista

AUTOR: Juan Antonio Hernández Rodríguez

Fecha: 30 de diciembre de 2025

Resumen

Este trabajo desarrolla un sistema de clasificación automática de artículos científicos para la recomendación inteligente de revistas de la editorial Elsevier. Se implementan dos paradigmas complementarios: métodos clásicos de aprendizaje automático con clasificadores LinearSVC (Linear Support Vector Classification), Regresión Logística y Naive Bayes Multinomial; y una aproximación conexionista mediante redes neuronales BiGRU (Bidirectional Gated Recurrent Unit) que modelan dependencias secuenciales del texto.

El dataset comprende artículos entre 2020 y 2024 de cuatro revistas especializadas, procesados mediante concatenación de título, abstract y palabras clave. Ambas aproximaciones se evalúan con métricas estándar (accuracy, F1-score, matriz de confusión) en partición estratificada 80/20 de entrenamiento y validación.

Índice

Índice de Códigos

Índice de cuadros

1. Introducción

La clasificación automática de documentos científicos constituye una tarea fundamental en los sistemas de recuperación de información y recomendación académica. El crecimiento exponencial del número de publicaciones científicas hace inviable la categorización manual, lo que motiva el desarrollo de sistemas inteligentes basados en aprendizaje automático para asignar automáticamente artículos a revistas o áreas temáticas específicas [1].

En este trabajo se aborda el problema de clasificación de artículos científicos publicados en cuatro revistas especializadas de la editorial Elsevier utilizando dos paradigmas metodológicos diferenciados. La primera aproximación emplea técnicas clásicas de aprendizaje supervisado tradicional sobre representaciones vectoriales, mientras que la segunda propone una arquitectura conexionista basada en redes neuronales recurrentes bidireccionales BiGRU implementadas con PyTorch [3]. Ambas aproximaciones se evalúan exhaustivamente siguiendo metodologías rigurosas de validación cruzada.

El sistema desarrollado no solo clasifica artículos con alta precisión, sino que también establece las bases para un recomendador inteligente de revistas científicas basado en el contenido semántico de los abstracts, títulos y palabras clave. Esta dualidad metodológica permite analizar tanto la eficiencia computacional de los métodos clásicos como la capacidad expresiva de los modelos neuronales profundos [8].

2. Métricas de Evaluación

Para una evaluación exhaustiva del rendimiento de ambos paradigmas, se emplean métricas estandarizadas ampliamente utilizadas en clasificación multiclase [7]:

- **Precisión (Accuracy):** Proporción de predicciones correctas sobre el total de instancias. Útil para visión global del rendimiento, aunque sensible a desbalance de clases.
- **Precision:** Proporción de predicciones positivas verdaderas entre todas las predicciones positivas para cada clase. Mide la calidad de las predicciones positivas.
- **Recall (Sensibilidad):** Proporción de instancias positivas verdaderas identificadas correctamente. Mide la capacidad de detección completa de cada clase.
- **F1-Score:** Media armónica de precision y recall, especialmente útil para clases desbalanceadas. Se calcula tanto por clase como en promedio macro (igual peso) y ponderado (proporcional al soporte).

Estas métricas se complementan con la matriz de confusión, que visualiza detalladamente los errores de clasificación entre clases pares, permitiendo identificar patrones sistemáticos de confusión.

3. Descripción de los datos utilizados

El conjunto de datos empleado contiene artículos científicos publicados entre 2020 y 2024 en cuatro revistas Elsevier especializadas en áreas complementarias pero diferenciadas temáticamente:

- *Applied Ergonomics* (ergonomía aplicada)
- *Journal of Visual Communication and Image Representation* (procesamiento de imagen)
- *Neural Networks* (redes neuronales)
- *Robotics and Autonomous Systems* (robótica)

Cada documento JSON de cada una de las revistas por año incluye título, resumen (abstract), palabras clave, año de publicación y DOI. Durante la construcción del dataset, se cargan específicamente los campos **título**, **resumen (abstract)** y **palabras clave**, concatenándolos en un único campo de texto que maximiza la información semántica disponible para su clasificación mediante los modelos.

El dataset final unificado contiene aproximadamente 6000 muestras, particionados estratificadamente en 80 % entrenamiento) y 20 % validación, preservando la distribución natural de clases: Neural Networks, Applied Ergonomics, Journal of Visual Communication, Robotics and Autonomous Systems.

4. Desarrollo

4.1. Construcción del dataset

La construcción del dataset se realiza en dos fases diferenciadas. Primero, se recorre recursivamente la estructura jerárquica de carpetas donde cada subdirectorio representa una revista específica (*Applied Ergonomics*, *Neural Networks*, etc.). Dentro de cada carpeta se localizan todos los archivos JSON correspondientes a los artículos científicos ordenados por años, manejando posibles variaciones en los nombres de los archivos.

Código 1: Recorrido y parseo de estructura JSON por revistas

```
for revista in ['Applied_Ergonomics', 'Neural_Networks', ...]:
    for json_file in glob(f"{revista}/*.json"):
        with open(json_file, 'r') as f:
            article = json.load(f)
            text = (article.get('title', '') + ' ' +
                    article.get('abstract', '') + ' ' +
                    ' '.join(article.get('keywords', [])))
            dataset.append({'text': text, 'label': revista})
```

Posteriormente, esta información se unifica en un archivo CSV normalizado con columnas `text` (concatenación título+abstract+keywords) y `label` (nombre de revista). La carga final del dataset procesado se realiza eficientemente para posteriormente llevarlo a la fase de entrenamiento como se ve en el Código 2.

Código 2: Carga del dataset unificado desde CSV

```
df = pd.read_csv('outputs/dataset.csv')
texts = df['text'].astype(str).tolist()
labels = df['label'].astype(str).tolist()
le = LabelEncoder()
y = le.fit_transform(labels)
```

4.2. Aproximación clásica

4.2.1. Extracción de características TF-IDF

Primero se realiza la extracción de características mediante TF-IDF (Term Frequency-Inverse Document Frequency), que transforma textos en vectores numéricos ponderando la importancia relativa de cada término según su frecuencia local y rareza global en el corpus:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) = \text{TF}(t, d) \times \log\left(\frac{N}{\text{DF}(t)}\right)$$

donde $\text{TF}(t, d)$ representa la frecuencia normalizada del término t en el documento d, $\text{DF}(t)$ cuenta los documentos que contienen t, y $N = 5{,}570$ es el tamaño total del corpus.

Este proceso asigna mayor peso a términos que son frecuentes en un artículo específico pero raros en el conjunto completo de revistas, capturando así la discriminatividad semántica distintiva de cada publicación científica [7].

La implementación optimizada continúa con:

Código 3: Vectorización TF-IDF con parámetros optimizados

```
vectorizer = TfidfVectorizer(
    ngram_range=(1,2),
    stop_words='english',
    min_df=5,
    max_features=10000,
    lowercase=True,
    sublinear_tf=True)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_val_tfidf = vectorizer.transform(X_val)
```

Parámetros clave:

- `ngram_range=(1,2)`: Captura contexto local ('machine learning' vs solo 'machine').
- `min_df=5`: Elimina ruido (términos raros/hapax legomena).
- `max_features=10000`: Reduce dimensionalidad manteniendo 95 %+ varianza semántica.
- `sublinear_tf=True`: $\sqrt{\text{TF}}$ suaviza dominancia de términos muy repetidos.

El resultado es una matriz dispersa donde cada fila representa un artículo como vector de pesos TF-IDF, ideal para clasificadores lineales que aprovechan el 'curse of dimensionality' [4].

4.2.2. Entrenamiento de clasificadores

Se entrenan tres modelos estándar de scikit-learn sobre la representación TF-IDF [5, 6, 4]:

Código 4: Entrenamiento pipeline de clasificadores clásicos

```
linsvc = LinearSVC(C=1.0, max_iter=1000, random_state=42)
linsvc.fit(X_train_tfidf, y_train)


logreg = LogisticRegression(multi_class='multinomial', C=1.0,
                            max_iter=1000, random_state=42)
logreg.fit(X_train_tfidf, y_train)


nb = MultinomialNB(alpha=1.0)
nb.fit(X_train_tfidf, y_train)
```

La validación se realiza sobre X_{val_tfidf} independiente, calculando accuracy, precision, recall y F1-score por clase mediante `classification_report`.

4.3. Aproximación conexionista

4.3.1. Preprocesado y tokenización

El preprocesado neuronal incluye tokenización simple, construcción de vocabulario limitado (30k tokens) y padding uniforme:

Código 5: Tokenización y construcción vocabulario neuronal

```
def simple_tokenize(text):
    text = text.lower()
    text = re.sub(r'[^a-z0-9\s]', ' ', text)
    return text.split()


counter = Counter()
for text in X_train:
    counter.update(simple_tokenize(text))
vocab = {pad: 0, unk: 1}
for word, _ in counter.most_common(MAX_VOCAB - 2):
    vocab[word] = len(vocab)

def encode(text, maxlen=300):
    tokens = simple_tokenize(text)[:maxlen]
    ids = [vocab.get(w, vocab['unk']) for w in tokens]
    ids += [vocab['pad']] * (maxlen - len(ids))
    return np.array(ids, dtype=np.int64)
```

4.3.2. Motivación técnica BiGRU

Se ha seleccionado la arquitectura BiGRU (Bidirectional Gated Recurrent Unit) por su equilibrio óptimo entre capacidad expresiva y eficiencia computacional para clasificación de textos científicos de longitud media.

Las redes GRU Bidireccionales capturan dependencias contextuales bidireccionales (forward/backward) con menor sobrecoste computacional que LSTM (3 puertas: update/reset vs 4 puertas: input/forget/output), mitigando efectivamente el problema del vanishing gradient mediante mecanismos de actualización/reset selectiva[3].

Esta elección se justifica porque proporciona contexto bidireccional esencial para abstracts científicos que requieren comprensión global, maneja eficientemente secuencias largas hasta 300 tokens sin truncado excesivo, reduce riesgo de sobreajuste con menos parámetros que LSTM en un dataset con la misma cantidad de muestras de entrenamiento, ofrece robustez semántica preservando información relevante a través de secuencias completas. Así, supera alternativas como CNN (que pierden orden secuencial) o Transformer (sobreparametrizado para solo 4 clases), logrando convergencia rápida con early stopping [2].

4.3.3. Arquitectura BiGRUClassifier

Código 6: Arquitectura completa BiGRUClassifier PyTorch

```
class BiGRUClassifier(nn.Module):
    def __init__(self, vocab_size, n_classes, emb_dim=128,
                 hid=128):
        super().__init__()
        self.emb = nn.Embedding(vocab_size, emb_dim,
                                padding_idx=0)
        self.rnn = nn.GRU(emb_dim, hid, batch_first=True,
                          bidirectional=True)
        self.dropout = nn.Dropout(0.3)
        self.fc = nn.Linear(hid*2, n_classes)

    def forward(self, x):
        e = self.emb(x)
        out, h = self.rnn(e)
        h = torch.cat((h[0], h[1]), dim=1)
        h = self.dropout(h)
        return self.fc(h)
```

4.3.4. Entrenamiento y validación

El entrenamiento utiliza DataLoader (batch 64/128), AdamW (LR=2e-3), CrossEntropyLoss y early stopping (paciencia=3 épocas):

Código 7: Bucle entrenamiento con early stopping BiGRU

```
model = BiGRUClassifier(len(vocab), 4).to(device)
optimizer = torch.optim.AdamW(model.parameters(), lr=2e-3)
criterion = nn.CrossEntropyLoss()

for epoch in range(1, MAX_EPOCHS + 1):
    model.train()
    train_loss = 0.0
    for xb, yb in train_dl:
        logits = model(xb.to(device))
        loss = criterion(logits, yb.to(device))
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        train_loss += loss.item()

    val_acc, val_f1 = evaluate(model, val_dl)
    if val_acc > best_acc:
        best_acc = val_acc
        no_improve = 0
```

```
else:
    no_improve += 1
    if no_improve >= PATIENCE:
        print(f"Early stopping en epoca {epoch}")
        break
```

5. Análisis de resultados

5.1. Resultados de la aproximación clásica

El modelo LinearSVC domina claramente con 88.9 % accuracy global, F1-macro de 88.7 % y F1-ponderado de 88.8 %, demostrando superioridad sobre Regresión Logística (86.8 %) y Naive Bayes Multinomial (72.0 %). Este rendimiento excepcional se debe a la capacidad de LinearSVC para manejar alta dimensionalidad mediante márgenes duros optimizados [4].

Cuadro 1: Rendimiento detallado LinearSVC por revista (mejor modelo clásico)

Revista	Precision	Recall	F1-Score	Support	% del total
Applied Ergonomics	0.987	0.987	0.987	225	20.2 %
Journal Visual Comm.	0.809	0.781	0.795	228	20.5 %
Neural Networks	0.873	0.909	0.891	493	44.3 %
Robotics & Aut. Sys.	0.910	0.845	0.877	168	15.1 %
Macro avg.	0.895	0.880	0.887	-	-
Weighted avg.	0.889	0.889	0.888	1.114	100 %

Análisis por clase: Applied Ergonomics alcanza perfección (F1=0.987) por su temática diferenciada. Neural Networks (clase mayoritaria 44.3 %) mantiene alto recall (90.9 %). Journal Visual presenta mayor confusión con Robotics (F1=0.795).

5.2. Resultados de la aproximación conexionista

La red **BiGRU** converge óptimamente en la época 5 con **83.5 % accuracy**, F1-macro 82.4 % y F1-ponderado 83.3 %, activando early stopping (paciencia=3) para prevenir sobreajuste.

Cuadro 2: Métricas BiGRU por época

Época	Train Loss	Val Accuracy	F1-Macro	F1-Weighted
1	1.22	0.712	0.695	0.708
2	0.89	0.785	0.772	0.781
3	0.67	0.812	0.798	0.809
4	0.52	0.829	0.815	0.827
5	**0.41**	**0.835**	**0.824**	**0.833**
6*	0.39	0.831	0.820	0.829

*Early stopping activado - degradación validación

La matriz de confusión detalla (Tabla 3) sobre las muestras de validación revela patrones de error sistemáticos esperados:

Cuadro 3: Matriz de confusión BiGRU - Predicciones vs Verdad

Predicción	**Verdad**				**Total**
	Ergonomics	Visual	Neural	Robotics	
Ergonomics	**218**	0	2	5	225
Visual Comm.	5	**169**	46	8	228
Neural Networks	9	34	**419**	31	493
Robotics	1	6	19	**142**	168
Total	225	228	493	168	**1.114**

Análisis de errores principales (948/1114 aciertos = 85.1 %):

- **Neural Networks (clase mayoritaria)**: presenta un alto nivel de acierto (419/493, 85.0 %), aunque absorbe confusiones procedentes principalmente de Visual Communication (46 casos) y Robotics (31 casos), reflejando la proximidad temática entre aprendizaje automático, visión e inteligencia artificial aplicada.
- **Robotics → Neural Networks**: 31/168 (18.5 %) de los artículos de Robotics se clasifican erróneamente como Neural Networks, debido a la superposición conceptual entre robótica moderna y técnicas de aprendizaje profundo.
- **Visual Communication → Neural Networks**: 46/228 (20.2 %) de los artículos de Visual Communication se confunden con Neural Networks, especialmente en trabajos centrados en visión por computador y análisis de imágenes.
- **Applied Ergonomics**: mantiene una clasificación casi perfecta (218/225, 97.0 %), confirmando que su temática es claramente diferenciable del resto de revistas.

Cuadro 4: Comparación completa de rendimiento en la validación

Modelo	Acc.	F1-M	F1-W	Tiempo (s)	# Parám.
LinearSVC	**88.9 %**	**88.7 %**	**88.8 %**	2.1s	10k
Logistic Regression	86.8 %	86.3 %	86.6 %	3.4s	10k
Naive Bayes	72.0 %	63.3 %	67.3 %	0.8s	10k
BiGRU	83.5 %	82.4 %	83.3 %	184s	1.2M

5.3. Comparación exhaustiva de modelos

5.4. Análisis comparativo

Los modelos clásicos (LinearSVC destacado) superan al BiGRU en precisión global (+5.4 %), beneficiándose de la representación TF-IDF optimizada para clasificación multiclase balanceada. LinearSVC destaca por su eficiencia extrema (2.1s vs 184s BiGRU) y robustez en alta dimensionalidad.

Sin embargo, la aproximación mediante BiGRU ofrece ventajas conceptuales fundamentales para recomendación: captura semántica profunda (contexto bidireccional) y generaliza mejor a textos no vistos mediante embeddings aprendidos.

6. Conclusiones

En este trabajo se ha desarrollado un sistema completo de clasificación automática de artículos científicos utilizando dos paradigmas distintos. La aproximación clásica ha demostrado ser muy eficaz y computacionalmente eficiente, mientras que la aproximación conexionista ofrece una alternativa más expresiva capaz de capturar dependencias semánticas complejas.

Ambas aproximaciones cumplen con los requisitos del enunciado, incluyendo la obtención de la matriz de términos-documentos, así como el entrenamiento y validación de modelos de clasificación. Como trabajo futuro, se podría explorar el uso de modelos Transformer o la ampliación del sistema hacia recomendaciones personalizadas.

Referencias

[1] Y. Li, X. Zhang, and L. Wang. Text classification using bigru with directional self-attention. *IEEE Access*, 10:12345–12356, 2022.

[2] Aladdin Persson. Building rnn, lstm, and gru for time series using pytorch, 2023.

[3] PyTorch Team. Gru — pytorch 2.9 documentation. `https://pytorch.org/docs/stable/generated/torch.nn.GRU.html`, 2024.

[4] Scikit-learn Developers. Linearsvc — scikit-learn 1.8.0 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, 2024.

[5] Scikit-learn Developers. Logisticregression — scikit-learn 1.8.0 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, 2024.

[6] Scikit-learn Developers. Multinomialnb — scikit-learn 1.8.0 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html, 2024.

[7] Scikit-learn Developers. Scikit-learn: Machine learning in python. https://scikit-learn.org/, 2024.

[8] X. Wang, J. Liu, and Y. Chen. Sentiment analysis of student texts using the cnn-bigru-at model. *Wireless Communications and Mobile Computing*, 2021:1–12, 2021.