PUCE Virtual

CLASE

8

Procesamiento de Lenguaje Natural

Integración y Comunicación de Resultados

QS WORLD UNIVERSITY RANKINGS

Educación de calidad

INTRODUCCIÓN DE LA CLASE

GRADO / POSGRADO / TECNOLOGÍAS

Estudia a tu tiempo
son interrupciones

EDUCACIÓN EN LÍNEA DE CALIDAD

Hasta la Semana 7, los contenidos, el material de refuerzo y los retos 1, 2 y 3 han construido progresivamente los componentes esenciales de un sistema de análisis textual para ciberseguridad: un dataset preparado y documentado, un clasificador de amenazas, un módulo de análisis semántico e intención maliciosa, y un esquema de validación con criterios de explicabilidad y ética. Sin embargo, estos elementos todavía pueden existir como piezas separadas. La Semana 8 introduce el paso final del curso: integrar todos los componentes en un sistema funcional de alerta temprana, capaz de apoyar la toma de decisiones en un entorno tipo SOC.

Esta etapa no consiste solo en programar un prototipo, sino en comprender cómo se articula un sistema real de PLN para ciberseguridad: cómo fluyen los datos, cómo se combinan los módulos de clasificación y semántica, cómo se evalúa el comportamiento global del sistema y cómo se comunica técnicamente su desempeño. Además, se incorpora una mirada integral sobre métricas avanzadas, sesgos, privacidad, explicabilidad y validación final. En otras palabras, la Semana 8 transforma lo aprendido en un sistema coherente, justificable y comunicable, cerrando el proyecto integrador del curso.

RDA 3: Evaluar sistemas de procesamiento de lenguaje natural para apoyar la toma de decisiones en ciberseguridad, a través de técnicas y métricas de precisión, eficacia, explicabilidad e impacto ético de los modelos.

Reto 4

Integración y Comunicación de Resultados.

Arquitectura de sistemas de PLN en ciberseguridad.

La arquitectura del sistema es la organización funcional de todos los módulos que participan en el procesamiento de texto y la generación de alertas. No se trata solo de "tener modelos", sino de definir:

- qué entra al sistema,
- qué se procesa en cada etapa,
- qué decisiones se toman,
- y qué sale como resultado.

En un entorno de ciberseguridad, esta arquitectura debe responder a una lógica operativa: recibir texto, analizarlo, priorizarlo y apoyar decisiones.

Componentes mínimos de una arquitectura PLN para SOC

Ilustración 1 muestra el pipeline del presente curso de Procesamiento de Lenguaje Natural.

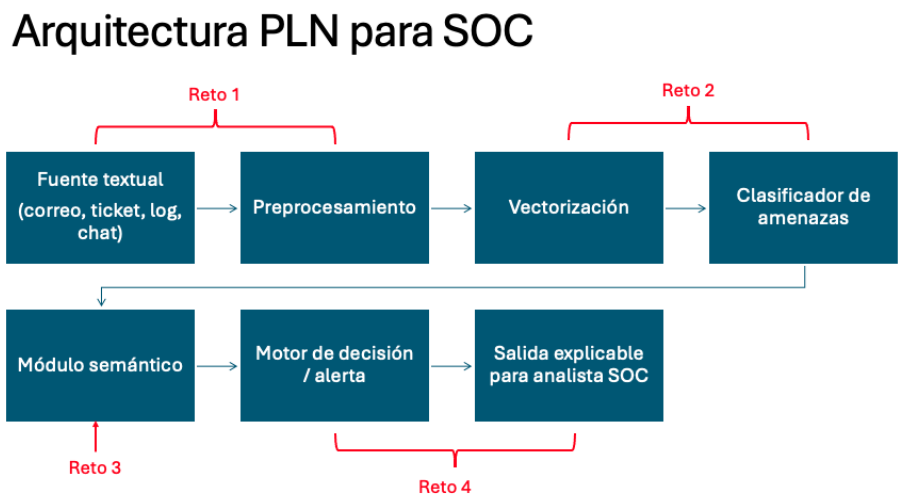

Ilustración 1. Arquitectura PLN para SOC (creación de autor Rafael Melgarejo)

Descripción de cada bloque:

- **Fuente textual:** es el punto de entrada del sistema, por ejemplo: correos sospechosos, tickets de soporte, mensajes corporativos, reportes narrativos.

- **Preprocesamiento:** transforma el texto crudo en texto analizable, incluye: tokenización, normalización, eliminación del ruido, lematización. Este componente proviene del reto 1.
- **Vectorización:** convierte el lenguaje en representaciones numéricas, utilizando: TD-IDF, embeddings, representaciones híbridas. Este paso permite alimentar los modelos posteriores.
- **Clasificador de amenazas**: estima si el texto pertenece a clases como: phishing, malware, ingeniería social, incidente técnico, legítimo. Corresponde al reto 2.
- **Modulo semántico:** permite detectar reformulaciones, identificar intención, analizar señales discursivas, y reconocer evasión semántica. Reto 3.
- **Motor de decisión:** integra resultados del clasificador y del módulo semántico. Ejemplo:
    - si clase = phishing y señal semántica alta → alerta crítica
    - si clase = legítimo pero anomalía alta → revisión humana
- **Salida explicable: emite alerta y la justifica. Ejemplo:**
    - Alerta: posible phishing.
    - Razones: urgencia + acción + autoridad + similitud con ataques previos.

Diferencia entre modelo y sistema

Aunque tanto el modelo como el sistema son tecnológicos, es decir sistemas cerrados (artificiales), en tecnología hay diferencia entre modelo y sistema, no son lo mismo.

MODELO	SISTEMA
Clasifica o detecta	Integra, decide y comunica
Produce una predicción	Produce una alerta útil
Puede funcionar aislado	Debe operar con flujo completo.

La arquitectura organiza varios modelos y reglas en un único flujo funcional.

Requisitos de una arquitectura útil en ciberseguridad

La arquitectura convierte componentes aislados en un sistema capaz de apoyar decisiones reales. Una arquitectura básica adecuada es:

- Modular: cada componente puede mejorarse por separado
- Explicable: las decisiones deben tomarse y justificarse
- Robusta: debe tolerar reformulación o evasión.
- Ética: debe controlar sesgos y prteger datos.

- Operativa: debe ser útil para analistas humanos.

La arquitectura del curso puede resumirse de la siguiente manera:

- Reto 1: prepara datos
- Reto 2: clasifica amenazas
- Reto 3: detecta intención maliciosa
- Semana 7: valida y controla éticamente
- Semana 8: integra todo en un sistema único.

Para ampliar el conocimiento sobre arquitectura, se recomienda: el video de ARIU " Subcomisión de Ciberseguridad CIN: Arquitectura SOC con Software Libre" en: https://www.youtube.com/watch?v=cUOs6ux6EsI

Ilustración 2 muestra una arquitectura de ejemplo en pseudocódigo

```
def sistema_alerta(texto):
    texto_limpio = preprocesar(texto)
    vector = vectorizar(texto_limpio)

    clase = clasificador.predict(vector)
    semantica = modulo_semantico(texto_limpio)
    explicacion = explicar_decision(texto_limpio)

    if clase == "phishing" or semantica == "ataque":
        return {
            "alerta": True,
            "nivel": "alto",
            "explicacion": explicacion
        }

    return {
        "alerta": False,
        "nivel": "bajo",
        "explicacion": explicacion
    }
```

Ilustración 2. Pseudocódigo de Arquitectura de PLN (creación de autor Rafael Melgarejo)

Integración clasificación + semántica.

Al momento, el sistema cuenta con dos capacidades distintas:

- Clasificador (Reto 2) → identifica qué tipo de amenaza es el texto
- Módulo semántico (Reto 3) → interpreta cómo está construido el mensaje (intención, evasión, señales discursivas)

Cada uno resuelve un problema diferente. Sin embargo, en un entorno real de ciberseguridad, ninguno es suficiente por sí solo:

- El clasificador puede fallar ante reformulaciones.
- El análisis semántico puede detectar intención, pero no clasificar el tipo de amenaza.

La integración permite construir un sistema más robusto, preciso y contextualizado.

Las ventajas de la integración pueden resumirse en:

- Detecta ataques explícitos y evasivos
- Reduce dependencia de palabras clave
- Mejora balance precision/recall
- Permite decisiones más inteligentes
- Facilita explicabilidad

Mientras, los riesgos si se integra mal, pueden ser:

- Exceso de falsos positivbos (modelo demasiado sensible)
- Pérdida de ataques reales (modelo demasiado estricto)
- Decisiones inconsistentes
- Falta de interpretabilidad

La clasificación dice qué es el mensaje, la semántica dice cómo está construido, y la integración decide qué hacer con él.

Tipos de integración (Eisenstein, 2019)

A continuación se ejemplifica las cuatro integraciones más conocidas:

Tipo de integración	Ejemplo	Aspecto Positivo	Aspecto Negativo
Básica (OR lógico)	Si legítimo o semántico, entonces ataque	Alta sensibilidad	Puede generar falsos positivos
Estricta (AND lógico)	Si legítimo y semántico, entonces ataque	Reduce falsos positivos	Puede perder ataques evasivos
Ponderada	Asigna peso a clasificación y semántica	Recomendada	Establecer la relación de pesos adecuada
Con Reglas contextuales	Integra anteriores	Equilibra Precision, Recall y Control ético	

Tipos de integración en Python

Ilustración 3 muestra un pseudocódigo que integra clasificación y semántica.

```python
def sistema_integrado(texto):
    texto_limpio = texto.lower()

    # Clasificador (simulado)
    if "contraseña" in texto_limpio:
        clase = "phishing"
    else:
        clase = "legitimo"

    # Semántica
    if "inmediatamente" in texto_limpio or "urgente" in texto_limpio:
        semantica = "ataque"
    else:
        semantica = "neutral"

    # Decisión
    if clase == "phishing" and semantica == "ataque":
        decision = "alerta_critica"
    elif semantica == "ataque":
        decision = "revision_humana"
    elif clase != "legitimo":
        decision = "alerta_media"
    else:
        decision = "sin_alerta"

    return {
        "clase": clase,
        "semantica": semantica,
        "decision": decision
    }
```

Ilustración 3. Pseudocódigo Integrado (creación de autor Rafael Melgarejo)

Integración con explicabilidad

El sistema también debe devolver el por qué, como el pseudocódigo de Ilustración 4.

```python
def sistema_con_explicacion(texto):
    resultado = sistema_integrado(texto)

    explicacion = []

    if "inmediatamente" in texto:
        explicacion.append("urgencia")

    if "contraseña" in texto:
        explicacion.append("dato sensible")

    resultado["explicacion"] = explicacion

    return resultado
```

Ilustración 4. Pseudocódigo explicabilidad (creación de autor Rafael Melgarejo)

Un ejemplo de salida final de los pseudocódigos presentados en Ilustración 3 y 4, es:

- Texto (input): “Actualice su contraseña inmediatamente”
- Output:

- Clase: phishing
- Semántica: ataque
- Decisión: [urgencia, dato sensible]

Métricas avanzadas: ROC, AUC (Jurafsky, 2023).

Es necesario evaluar el sistema ya integrado, no solamente el modelo individual.

Las métricas como: Precision, Recall y F1-score son útiles, pero tienen una limitación importante: dependen de un umbral fijo de decisión.

En ciberseguridad, esto es crítico porque:

- el mismo sistema puede comportarse muy distinto según el umbral,
- no siempre se quiere el mismo equilibrio entre falsos positivos y falsos negativos,
- las decisiones deben adaptarse al nivel de riesgo del entorno (SOC).

Por eso, existen métricas que permiten evaluar el sistema a lo largo de múltiples umbrales, no solo en uno.

Curva ROC (Receiver Operating Characteristic)

La curva ROC muestra cómo cambia el comportamiento del modelo al variar el umbral de decisión. ROC no mide qué tan buena es la decisión, sino qué también se puede decidir si se cambia el umbral. Los ejes son:

- Eje X → FPR (False Positive Rate)
- Eje Y → TPR (True Positive Rate = Recall)

Interpretación:

Cada punto de la curva representa un umbral distinto.

- Umbral bajo → detecta más ataques (alto recall), pero más falsos positivos
- Umbral alto → menos falsos positivos, pero puede perder ataques

ROC mide la capacidad discriminativa del modelo, es decir la manera en que el modelo separa las clases.

AUC (Area Under the Curve – Área Bajo la Curva)

AUC es la probabilidad de que el modelo calcule un ataque por encima de un contexto legítimo. Los valores típicos son:

- 0.5 aleatorio
- 0.7 a 0.8 aceptable
- 0.8 a 0.9 bueno
- >0.9 excelente

Por ejemplo, si un modelo evalúa correos: si AUC=0.92 significa muy buen separador; si AUC=0.65, entonces separación débil.

Ilustración 5 muestra un ejemplo en Python, que supone existe un modelo "model" ya entrenado.

```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

#Obtener probabilidades
y_scores = modelo.predict_proba(X_test)[:, 1]

#Calcular ROC
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)

#Graficar
plt.figure()
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
plt.plot([0,1], [0,1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Curva ROC")
plt.legend()
plt.show()
```

Ilustración 5. Cálculo y graficación de ROC (creación de autor Rafael Melgarejo)

Relación con SOC

La curva ROC permite elegir el umbral óptimo según el contexto

Escenario	**Estrategia**
Alta seguridad	Umbral bajo → más detección
Alta precisión operativa	Umbral alto → menos falsos positivos

Usualmente los datos están desbalanceados, en este caso ROC se complementa con Precision-Recall. El video "CURVAS ROC Y ÁREA BAJO LA CURVA (AUC) | #34 Curso Machine Learning con Python" de AprendeIA con Ligdi Gonzalez en https://www.youtube.com/watch?v=AcbbkCL0dlo .

Sesgos, privacidad y explicabilidad.

Si un sistema no controla sesgos, protege datos y explica sus decisiones, no es confiable. Para pasar de la dimensión técnica a la dimensión responsable del sistema, hay que preguntarse si el sistema funcional (puede ser utilizado en un entorno real de ciberseguridad):

- ¿es justo? (sesgos)
- ¿respeta la información? (privacidad)
- ¿se puede confiar en sus decisiones? (explicabilidad)

Sesgos

Ocurre cuando el modelo: favorece o penaliza ciertos patrones lingüísticos, aprende asociaciones incorrectas, o generaliza de forma injusta a partir de ciertos datos. Por ejemplo, si el dataset tiene muchos ataques con lenguaje urgente, el mensaje puede aprender que todo mensaje urgente es un ataque, generado: falsos positivos en comunicaciones legítimas, y decisiones injustas o incorrectas. A continuación tipos de sesgos y la manera de mitigarlos sugerida (Jurafsky,2023).

SESGO	MITIGACIÓN
Lingüístico (asociado a estilo o tono)	Usar datasets diversos
De Clase (por desbalance de datos)	Balancear clases
De Contexto (por dominio limitado del dataset)	Evaluar casos ambiguos, validar con ejemplos reales

Privacidad en análisis de texto

Los sistemas de PLN en ciberseguridad pueden analizar: correos electrónicos, tickets internos, chats corporativos, reportes técnicos. Todos los anteriores pueden contener: nombres, credenciales e información sensible.

Los riesgos por privacidad pueden ser (Cavoukian,2010): exposición de datos confidenciales, almacenamiento y uso indebido de información personal.

Los principios de privacidad en PLN incluyen: minimización de datos (usar solo lo necesario), anonimización (eliminar identificadores), y seguridad de almacenamiento (proteger datasets).

Un ejemplo conceptual práctico de formas de anonimización

```python
import re

def anonimizar(texto):
    texto = re.sub(r'\S+@\S+', '[EMAIL]', texto)
    texto = re.sub(r'\d{10}', '[NUMERO]', texto)
    return texto
```

Ilustración 6. Anonimización de datasets (creación de autor Rafael Melgarejo)

Explicabilidad (en contexto operativo) (Eisenstein, 2019)

Es un requisito del sistema final que debe explicar:

- La manera en que detecta una amenaza
- La señales que encontró
- Su nivel de confianza
- El módulo influyente (clasificador o semántico)

Si el sistema no explica lo anterior, corre el riesgo de que el analista pierda confianza, no se pueda auditar decisiones, y rechazo del sistema en entorno real.

Ilustración 7 ejemplifica en pseudocódigo un sistema confiable.

```
def sistema_responsable(texto):
    texto = anonimizar(texto)

    resultado = sistema_integrado(texto)
    explicacion = resultado.get("explicacion", [])

    # control de sesgo simple
    if "urgencia" in explicacion and len(explicacion) == 1:
        decision = "revision_humana"
    else:
        decision = resultado["decision"]

    return {
        "decision": decision,
        "explicacion": explicacion
    }
```

Ilustración 7. Pseudocódigo sistema responsable (creación de autor Rafael Melgarejo)

Validación técnica final.

Es la evaluación integral del sistema completo, no de componentes aislados. En esta etapa se verifica si el sistema:

- funciona correctamente de extremo a extremo,
- cumple con los objetivos del curso,
- es consistente en distintos escenarios,
- y es suficientemente confiable para apoyar decisiones en ciberseguridad.

Ya no se evalúa solo: modelo → resultado, sino:

entrada real → procesamiento → integración → decisión → explicación

Dimensiones de validación del sistema completo

- Dataset incluye (Ilustración 8 con ejemplo)
    - Casos reales o simulados
    - Ataques explícitos
    - Ataques evasivos
    - Textos legítimos
    - Casos ambiguos

```
dataset_final = [
    ("Actualice su contraseña inmediatamente", "ataque"),
    ("Se recomienda validar su acceso", "ataque"),
    ("Reunión del equipo mañana", "legitimo"),
    ("Revise el documento cuando pueda", "legitimo"),
    ("Confirme su cuenta para evitar interrupciones", "ataque")
]
```

Ilustración 8. Dataset con todas las opciones (creación de autor Rafael Melgarejo)

- Correctitud funcional:
    - Si el sistema procesa correctamente los textos
    - Si los módulos están bien conectados
- Desempeño global:
    - Precision, Recall, F1
    - ROC / AUC
    - Comportamiento ante distintos umbrales
- Robustez
    - Funcionamiento con textos reformulados
    - Resistencia a evasión semántica
    - Desempeño frente a casos ambiguos
- Explicabilidad
    - Justificación de decisiones
    - Coherencia de la explicación
- Ética y control
    - Evita decisiones injustifficadas
    - Aplica revisión humana cuando corresponde.

Evaluación

- Evaluación automática (Ilustración 8), que incluye métricas globales.

```
#Evaluación sistema completo
import pandas as pd

df = pd.DataFrame(dataset_final, columns=["texto", "real"])

df["pred"] = df["texto"].apply(lambda x: sistema_responsable(x)["decision"])
df

#Métricas globales
from sklearn.metrics import classification_report

print(classification_report(df["real"], df["pred"]))
```

Ilustración 9. Evaluación sistema PLN (creación de autor Rafael Melgarejo)

- Evaluación cualitativa, se debe analizar
    - ¿Dónde falla el sistema?
    - ¿Qué errores comete?
    - ¿Son errores críticos (seguridad) o tolerables?
- Pruebas de estrés (stress testing)
    - Reformulación: “Valide su acceso para mantener operatividad”
    - Ambigüedad: “Sería bueno revisar esto pronto”
    - Ruido: “*** urgente!! Validar cuenta ahora!!! ***”
- Validación de consistencia

- Estable: no cambiar decisiones sin razón
- Coherente: decisiones alineadas con lógica definida
- Reproducible: mismos resultados con mismos datos

- Validación de umbrales
  - Niveles de alerta del sistema: bajo, medio, alto.
  - Verificar
    - Si son adecuador
    - Si hay muchas alertas
    - Si se pierden amenazas reales

Comunicación de resultados y decisiones.

En un entorno de ciberseguridad, el sistema no "termina" cuando genera una predicción. Su valor real depende de cómo comunica los resultados a los analistas humanos. Un modelo altamente preciso puede ser inútil si:

- no presenta la información de forma clara,
- no prioriza correctamente las alertas,
- no explica sus decisiones.

Por ello, la última etapa del sistema consiste en transformar los outputs técnicos en información accionable, es decir, en insumos que permitan tomar decisiones operativas dentro de un SOC.

Entregables del sistema completo

- Clasificación del evento: phishing / malware / ingeniería social / legítimo
- Nivel de riesgo: alto / medio / bajo
- Explicación: señales detectadas (urgencia, autoridad, acción) / similitud semántica / factores de decisión.
- Recomendación operativa: bloquear mensaje / escalar a analista / monitorear / permitir.
- Ejemplo de salida completa
  - ALERTA: phishing
  - NIVEL: alto
  - EXPLICACIÓN
    - Urgencia detectada ("inmediatamente")
    - Solicitud de acción ("verifique su cuenta")
    - Similitud con ataques previos
  - RECOMENDACIÓN: Escalar a analista SOC y bloquear remitente.

Diseño de la salida del sistema

Estructuración de la información en forma directa, clara y jerárquica, como indica el pseudocódigo de la Ilustración 10.

```
def salida_sistema(texto):
    resultado = sistema_responsable(texto)

    return {
        "texto": texto,
        "decision": resultado["decision"],
        "nivel": "alto" if resultado["decision"] == "alerta_critica" else "medio",
        "explicacion": resultado["explicacion"],
        "recomendacion": "revisión SOC"
    }
```

Ilustración 10. Salida del Sistema Completo (creación de autor Rafael Melgarejo)

Principios de comunicación efectiva

La salida del sistema observa:

- Claridad: evitar términos técnicos innecesarios
- Prioridad: destacar alertas críticas primero
- Trazabilidad: permitir entender cómo se llegó a la decisión
- Accionabilidad: siempre incluir una recomendación.

Validación de la comunicación:

Evaluar si la información es comprensible, las explicaciones son suficientes, la recomendación es coherente, y si facilita la toma de decisiones.

El sistema completo tiene los siguientes componentes:

Datos → Preprocesamiento → Clasificación → Semántica → Evaluación → Ética → Comunicación → Decisión

Error común

Un error frecuente es general salidas muy técnicas e incomprensibles como: “Predicción: 0.873”. Esto es inútil para un SOC. Debe transformarse en: “Alerta de alta riesgo con justificación clara”.

El objetivo del PLN en ciberseguridad no es solo analizar texto, sino convertir lenguaje en decisiones informadas.

Glosario:

- **Cavoukian, A. (2010).** Privacy by Design: The 7 Foundational Principles. Information and Privacy Commissioner of Ontario.
- **Eisenstein, J. (2019).** Introduction to natural language processing. MIT Press. https://cdn.jsdelivr.net/gh/it-ebooks-0/it-ebooks-2018-04to07/Natural%20Language%20Processing%20%28Jacob%20Eisenstein%29.pdf
- **Jurafsky, D., & Martin, J. H. (2023).** Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (3rd ed., draft). Stanford University. https://web.stanford.edu/~jurafsky/slp3/.

PUCE Virtual

síguenos en:

www.pucevirtual.puce.edu.ec