PUCE Virtual

CLASE

2

PROCESAMIENTO DEL LENGUAJE NATURAL

Preparación del texto para análisis

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

Esta clase va desde la comprensión del texto no estructurado (Semana 1) hacia la transformación procedimental del lenguaje natural en una forma computable. En ciberseguridad, los correos, tickets y logs narrativos contienen ruido (firmas, HTML, URLs, cadenas reenviadas) y variaciones lingüísticas que, si no se gestionan, degradan la calidad del análisis y generan errores aguas abajo (por ejemplo, patrones falsos o vocabularios inflados). Por ello, esta semana consolida un flujo de trabajo básico de PLN aplicado: tokenización, normalización y limpieza, como precondición para cualquier extracción de patrones o entrenamiento de modelos.

En paralelo, la semana introduce el análisis exploratorio del texto para identificar distribuciones léxicas, frecuencias y regularidades lingüísticas vinculadas a amenazas, y presenta el principio de representación del lenguaje: la idea de que el texto debe mapearse a representaciones (p. ej., listas de tokens, n-gramas, vectores) que preserven señal útil para tareas posteriores. Este puente entre limpieza y exploración permite comenzar a conectar patrones lingüísticos con ataques, dejando listo el dataset del Reto 1 para etapas subsiguientes (etiquetado y clasificación en Reto 2)

Preparación del texto para análisis

Tokenización y normalización.

La tokenización y la normalización son importantes en ciberseguridad porque: reducen ruido, disminuyen dimensionalidad, mejoran el análisis exploratorio, preparan el corpus para representación vectorial (Tema 2.5).

La tokenización consiste en dividir el texto en fragmentos manejables (tokens), generalmente palabras. Esto permite que el sistema deje de "ver" oraciones completas y comience a operar sobre listas de unidades lingüísticas. Hay varios tipos de tokenización:

- **Tokenización básica por espacios:** Divide por espacios. Es simple pero no siempre precisa.
- **Tokenización basada en expresiones regulares:** Permite eliminar puntuación y separar correctamente palabras.
- **Tokenización avanzada (bibliotecas NLP):** Herramientas como spaCy o NLTK consideran reglas lingüísticas más complejas.

La normalización busca reducir variaciones innecesarias que no aportan significado.

Incluye:

- Conversión a minúsculas
- Eliminación de caracteres especiales
- Eliminación o reemplazo de URLs
- Eliminación de espacios múltiples.

Ejemplo con Python

Ilustración señala el código Python paso a paso.

```python
#Ejemplo de tokenización y normalización de un texto

import re #Módulo "re" es una herramienta para procesar expresiones regulares de texto

#texto original
texto= "Para evitar la suspensión INMEDIATA, confirme SUS credenciales aquí: http://banca_segura.com"

#Convertir a minúsculas todo el texto
texto = texto.lower()

#Eliminar signos de puntuación
texto = re.sub(r'[^\w\s]', '', texto)

#Eliminación de URL
texto = re.sub(r'http\S+', '[URL]', texto)

#Dividir en tokens
tokens = texto.split()
print(tokens)
```

```
['para', 'evitar', 'la', 'suspensión', 'inmediata', 'confirme', 'sus', 'credenciales', 'aquí', '[URL]']
```

Ilustración 1. Código Python para tokenización y normalización (Creación del autor: Rafael Melgarejo)

Es recomendable mantener el token “[URL]” para conservar la información estructural, evitar sobreajuste del texto, proteger privacidad y reducir la dimensionalidad. Los Grandes Modelos de Lenguaje realizan tokenización para procesar el texto (prompt) que introduce el usuario. Para conocer más sobre la tokenización en LLM se recomienda el video “¿Qué son los TOKENS? | Grandes Modelos de Lenguaje” disponible en https://youtu.be/p3cPzA4S_wk?si=mHBKoh2zVny06T4l

Eliminación de ruido: firmas, HTML, URLs.

En ciberseguridad, gran parte del texto recibido por un SOC no aporta significado semántico relevante para detectar amenazas.

Los principales tipos de amenazas y su localización en contenido son:

- **Estructural:** HTML, encabezados, respuestas encadenadas
- **Administrativo:** Firmas corporativas, avisos legales
- **Técnico redundante:** logs repetidos, timestamps duplicados

Este contenido introduce variabilidad artificial en el corpus y dificulta el aprendizaje automático. El objetivo no es borrar información, sino eliminar redundancia no lingüística.

Principio clave en ciberseguridad

No todo lo repetitivo es ruido (Jurafsky,2023). Una URL, un dominio externo, palabras urgentes NO constituyen ruido. El ruido es información que no cambia la intención comunicativa. Una limpieza excesiva destruye patrones discursivos del ataque, por eso no es recomendable: eliminar mayúsculas urgentes, borrar imperativos, eliminar enlaces.

Ejemplo de eliminación de ruido con Python

- Supongamos el siguiente texto:

```
Estimado usuario,

Confirme su cuenta inmediatamente:
http://secure-login.com

Saludos cordiales,
Departamento de Seguridad TI

Este mensaje es confidencial...
```

Ilustración 2. Texto para análisis de Ruido (creación de autor Rafael Melgarejo)

- **Problemas detectados:**

Fragmento	Tipo de ruido
Saludos cordiales	firma
mensaje confidencial	disclaimer

formato email	estructura repetitiva

- Eliminación de ruido paso a paso (código Python)

```python
#Eliminar HTML
import re

def remove_html(text):
    return re.sub(r'<.*?>', '', text)

#Eliminar firmas comunes
def remove_signature(text):
    return re.sub(r'(saludos|atentamente).*', '', text, flags=re.IGNORECASE)

#Eliminar disclaimers
def remove_disclaimer(text):
    return re.sub(r'este mensaje.*', '', text, flags=re.IGNORECASE)
```

Ilustración 3. Ejemplo Python eliminación de ruido (creación de autor Rafael Melgarejo)

Manejo de stopwords y lematización.

Después de tokenizar, normalizar y eliminar ruido, el texto aún contiene palabras que no aportan información semántica relevante para detectar amezanas. Por ejemplo en: “confirme su cuenta inmediatamente para evitar suspensión”, hay palabras que aparecen usualmente: “su” y “para”. Estas palabras aumentan el tamaño del vocabulario sin mejorar la detección. Entonces es necesario eliminar los stopwords y realizar una reducción lingüística.

Stopwords

Las stopwords son palabras funcionales muy frecuentes cuyo valor discriminativo es bajo, en español por ejemplo: el, la, los, las, de, en, para, su, sus, y, o. Sin embargo, en ciberseguridad no deben eliminarse todas las stopwords (e.g. “su cuenta”, “su contraseña”) porque pueden indicar personalización del ataque (Eisenstein,2019). Usualmente se usan listas de stopwords adaptadas al dominio.

Implementación en Python de stopwords.

NLTK es un kit de herramientas de Python que trabajan con datos de lenguaje natural. Proporciona interfaces para más de 50 corpus y recursos léxicos como WordNet, además de un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, lematización, etiquetado, análisis sintáctico y razonamiento semántico, contenedores para bibliotecas de PLN. Posee corpus de varios idiomas.

En las siguientes líneas de código se observa la importación de stopwords desde un corpus de español. Luego, coteja con los stopwords los tokens del texto original para eliminarlos si coinciden.

```python
from nltk.corpus import stopwords

stop_words = set(stopwords.words('spanish'))

def remove_stopwords(tokens):
    return [t for t in tokens if t not in stop_words]
```

Ilustración 4. Ejemplo Python stopwords (creación de autor Rafael Melgarejo)

Lematización

La lematización (o lemmatization en inglés) es un proceso lingüístico y computacional que consiste en reducir una palabra en su forma flexionada (es decir, conjugada, en plural, en femenino, con género, tiempo verbal, etc.) a su forma base o canónica, llamada lema. El lema es la forma que aparece en un diccionario, normalmente el infinitivo en verbos, el singular masculino en adjetivos/sustantivos, etc.. Prácticas comunes incluyen transformar los verbos conjugados a su forma regular. Ejemplo: de “confirmó” a “confirmar”, de “verificando” a “verificar”.

La lematización se usa en PLN para:

- Simplificar textos antes de análisis (búsquedas, clasificación de textos, chatbots, análisis de sentimientos, etc.).
- Que el ordenador trate como la misma palabra todas sus variantes (mejora la precisión en motores de búsqueda, recomendadores, etc.).
- Reducir el vocabulario y el ruido en modelos de machine learning.

Lematización en Python con spaCy.

spaCy es una biblioteca para el procesamiento avanzado del lenguaje natural en Python y Cython. Incluye pipelines preentrenados. Admite la tokenización y el entrenamiento en +70 idiomas. Tiene modelos de velocidad y redes neuronales de vanguardia para etiquetado, análisis sintáctico, reconocimiento de entidades con nombre, clasificación de texto y más, aprendizaje multitarea con transformadores preentrenados como BERT, así como un sistema de entrenamiento listo para producción y un empaquetado de modelos, implementación y gestión de flujos de trabajo sencillos. spaCy es un software comercial de código abierto, publicado bajo la licencia MIT.

Para usar spaCy con Python es necesario instalarlo primero, y para manejar vocabulario, sintaxis, entidades en español es común utilizar el modelo de procesamiento “es_core_news_sm”. La ilustración 5 muestra alternativas de instalación de modelos.

Advertencia: si utiliza el código en ambiente de Google Colab añada el signo de admiración de la siguiente manera: “!pip install -U spacy”.

```
pip install -U spacy
python -m spacy download es_core_news_sm   # modelo pequeño (~40 MB)
# O mejor precisión:
# python -m spacy download es_core_news_md   # mediano
# python -m spacy download es_core_news_lg   # grande (mejor lematización)
```

Ilustración 5. Instalación de spaCy y modelo de procesamiento (creación de autor Rafael Melgarejo)

La Ilustración 6 es código para lematizar palabras / frases con spaCy y el modelo pequeño "es_core_news_sm". Se incluye un texto demostrativo.

```
import spacy

# Cargar el modelo (usa 'es_core_news_sm' o 'md'/'lg' para mejor calidad)
nlp = spacy.load("es_core_news_sm")

texto = "Los niños corrían rápidamente por el parque mientras jugaban y reían."

# Procesar el texto
doc = nlp(texto)

# Mostrar cada token, su lema y su categoría gramatical (POS)
print("Token → Lema → POS")
for token in doc:
    print(f"{token.text:12} → {token.lemma_:10} → {token.pos_}")
```

Ilustración 6. Lematización de un texto con spyCy (creación de autor Rafael Melgarejo)

El resultado de la ejecución del código se presenta a continuación(Token, Lema y la categoría gramatical de los elementos del texto:

Token	Lema	POS
Los	el	DET
niños	niño	NOUN
corrían	correr	VERB
rápidamente	rápidamente	ADV
por	por	ADP
el	el	DET
parque	parque	NOUN
mientras	mientras	SCONJ
jugaban	jugar	VERB
y	y	CCONJ
reían	reir	VERB
.	.	PUNCT

Cuando el texto tiene verbos irregulares y formas muy flexionadas hay que tomar en cuenta las formas del lenguaje. En inglés el resultado de la lematización puede no sorprender al usuario final, pero en español es variado. El código de Ilustración 7 toma varias frases y las lematiza.

```
import spacy

nlp = spacy.load("es_core_news_sm")

frases = [
    "Fui al médico porque me dolía mucho la cabeza.",
    "Estuvimos estudiando toda la noche y nos quedamos dormidos.",
    "Las mejores casas están siendo construidas ahora mismo.",
    "¡Corre, corrió, corriendo, correremos!"
]

for frase in frases:
    doc = nlp(frase)
    lemas = [token.lemma_ for token in doc]
    print(f"Original: {frase}")
    print(f"Lemas:    {' '.join(lemas)}\n")
```

Ilustración 7. Lematización verbos irregulares (creación de autor Rafael Melgarejo)

El resultado resulta curioso para quien no está acostumbrado:

Original: Fui al médico porque me dolía mucho la cabeza.

Lemas: ir al médico porque yo doler mucho el cabeza .

Original: Estuvimos estudiando toda la noche y nos quedamos dormidos.

Lemas: estuvir estudiar todo el noche y yo quedar dormido .

Original: Las mejores casas están siendo construidas ahora mismo.

Lemas: el mejor casa estar ser construir ahora mismo .

Original: ¡Corre, corrió, corriendo, correremos!

Lemas: ¡ Corre , correr , correr , correremo !

Como se puede observar, con el código proporcionado, el lema de “correremos” es “correremo”, que no es del todo correcto. spaCy utiliza el modo “rule”, y para corregir se lo trata explícitamente. En Ilustración 8 está el código inicial y en la parte inferior el resultado.

```python
import spacy

nlp = spacy.load("es_core_news_sm")

# Ver el componente lematizador
lemmatizer = nlp.get_pipe("lemmatizer")
print(lemmatizer.mode)  # Normalmente 'rule'

# Lematizar una sola palabra manualmente (raro, pero posible)
# La forma correcta es procesar el texto con el objeto nlp y luego obtener el lema del token.
word_to_lemmatize = "corrieron"
doc = nlp(word_to_lemmatize)

# Asumiendo que 'corrieron' es un solo token, accedemos a su lema
if doc:
    print(f"Lema de '{word_to_lemmatize}': {doc[0].lemma_}")
```

```
rule
Lema de 'corrieron': correr
```

Ilustración 8. Lematización explícita (creación de autor Rafael Melgarejo)

Se recomienda el video tutorial “NLP - 3 – Lematización” para revisar casos prácticos mediante Jupyter. El video está en https://youtu.be/YR-AGoq5K-M?si=FibS4_xd2TWJLs31
Con los pasos realizados hasta aquí, el texto ya no es solo lenguaje, son datos analizables, es decir se ha construido el pipeline de PLN:
TEXTO CRUDO→Tokenización→Normalización→Eliminación de ruido → Stopwords→Lematización→CORPUS LIMPIO.

Distribución léxica y frecuencias.

Con el objetivo de detectar regularidades discursivas, la distribución léxica describe: qué palabras aparecen, cuántas veces aparecen, y cómo se distribuyen en el corpus.
En ciberseguridad, un corpus de phishing (por ejemplo), suele mostrar: verbos imperativos frecuentes, referencias a cuentas o accesos, marcadores de urgencia. Mientras que un corpus legítimo presenta: descripciones técnicas, mensajes de error, y lenguaje colaborativo.

Ley de Zipf

En lenguaje natural pocas palabras aparecen muchas veces, y muchas palabras aparecen pocas veces, esto es distribución de Zipf. En otras palabras, el vocabulario crece, pero solo un subconjunto concentra la información útil.

Implementación de distribución léxica y frecuencia en Python

La Ilustración 9 presenta el código Python para lematizar el texto, calcular las frecuencias usando “collections” y las visualiza con “matplotlib”.

```python
import spacy

nlp = spacy.load("es_core_news_sm")

#función para lematizar el texto
def lemmatize(text):
    doc = nlp(text)
    return [token.lemma_ for token in doc]

#crea nueva columna lemmas
df["lemmas"] = df["texto_limpio"].apply(lemmatize)
# lemmas representa el lenguaje reducido a sus unidades semánticas base.

all_tokens = []

for lemmas in df["lemmas"]:
    all_tokens.extend(lemmas)

#Cálculo de frecuencias paso a paso

#Unificar tokens
all_tokens = sum(df["lemmas"], [])

#Contar ocurrencias
from collections import Counter

freq = Counter(all_tokens)
freq.most_common(20)

# Para visualizar las ocurrencias
import matplotlib.pyplot as plt

words, counts = zip(*freq.most_common(15))

plt.bar(words, counts)
plt.xticks(rotation=45)
plt.title("Frecuencia léxica del corpus")
plt.show()
```

Ilustración 9. Cálculo y Visualización de frecuencias con Python (creación de autor Rafael Melgarejo)

Principio de representación del lenguaje.

Representación del lenguaje es la manera de convertir el lenguaje humano en números sin perder significado. Las computadoras no entienden palabras, así un modelo de parendizaje automático solamente puede operar con: números, vectores y matrices. En consecuencia, cada documento se transforma en un vector numérico que describe su contenido lingüístico. Un texto, por ejemplo “confirme su cuenta inmediatamente” se convierte en: “[0,1,0,3,0,2…]. Este vector representa: las palabras que aparecen, con qué frecuencia, y su importancia dentro del texto. A continuación dos modelos de representación del lenguaje: Bag of Words, N-grams, y TF-IDF (Jurafsky,2023) .

BoW: Bag of Words

BoW permite detectar patrones como presencia de palabras críticas, vocabulario típico de phishing, y diferencias entre textos legítimos y maliciosos. BoW ignora el orden lenguaje, solo registra presencia y frecuencia de palabras clave. Aunque BoW trata igual palabras muy comunes y palabras importantes; ejemplo: cuenta (muy frecuente)y urgente (más

discriminativa). Aunque BoW es simple, interpretable e ideal para enseñanza inicial, no entiende significado ni contexto.

En Python es útil la librería “sklearn”, una herramienta predictiva con Machine Learning para el análisis de datos, especialmente textos, a los cuales transforma en un vector con las veces que aparece cada palabra de un documento.

N-grams (extension de BoW)

Mientras BoW ignora orden, n-grams recupera fragmentos lingüísticos. Esto es importante en ciberseguridad, porque los ataques vienen en combinaciones, no en palabran aisladas. En PLN es posible construir los n-gramas sobre la base de distintos tipos de elementos como por ejemplo fonemas, sílabas, letras, palabras.

TF-IDF (Term Frequency-Inverse Document Frequency)

Es una medida estadística que evalúa la importancia de una palabra dentro de un documento en relación con una colección (corpus). Combina la frecuencia del término (TF) con su rareza (IDF) para resaltar palabras clave significativas, reduciendo el peso de palabras comunes (ej. "el", "que"). La idea central es que no todas las palabras pesan igual, TF-IDF aumenta el peso de las palabras frecuentes en un documento y a su vez raras en el corpus.

En Python con sklearn el vector resultante representa importancia semántica, más que conteo.

Relación entre patrones lingüísticos y ataques.

Un patrón lingüístico es una regularidad observable en el lenguaje que aparece repetidamente en textos maliciosos. Por ejemplo, en este conjunto de señales “urgente + verificar + cuenta + [URL]” rara vez aparece en comunicaciones técnicas legítimas.

Dimensión discursiva del ataque

Los ataques de ingeniería social operan en tres niveles, como consta a continuación:

Nivel	Manifestación
Cognitivo	activar miedo, obediencia o urgencia
Discursivo	uso estratégico del lenguaje
Computacional	patrones detectables por PLN

El modelo aprende el tercer nivel a partir de los dos primeros (cognitivo y discursivo).

Ejemplos comparativos

- **En el texto:** “URGENTE: confirme su cuenta inmediatamente para evitar suspensión”, los patrones detectables son:
    - Imperativo → confirme
    - Urgencia → URGENTE, inmediatamente

- Amenaza → suspensión
- Acción → [URL]

- **En el texto:** "No puedo acceder a la VPN desde ayer. Aparece error de autenticación", los patrones detectables son:
  - Descripción técnica
  - Narrativa explicativa
  - Ausencia de coerción.

Como se puede concluir la diferencia central está en la intención comunicativa.

Patrones básicos

El objetivo de un sistema de PLN en ciberseguridad es detectar patrones discursivos asociados a ataques, no es detectar hackers. La ciberseguridad también es análisis del lenguaje. Los patrones que se identifican usualmente son:

- **Léxicos:** palabras específicas como: urgente, verificar, credenciales.
- **Sintácticos:** estructuras comunes como: verbos imperativos, o frases cortas de acción.
- **Semánticos:** la intención implícita como : presión temporal, suplantación institucional, manipulación emocional.

Las principales formas en que se relacionan están en Tabla 1.

Tabla 1. Patrones lingüísticos en ciberseguridad con PLN. Adaptado de (Jurafsky,2023)

Tipo de ataque	Patrones lingüísticos típicos que delatan al atacante	Cómo los detecta el PLN
Phishing / Spear-phishing	Urgencia ("actúa ya", "cuenta suspendida"), errores gramaticales, tono manipulador, palabras clave emocionales, inconsistencias sintácticas	Análisis de sentimiento, n-gramas, estilometría, embeddings (BERT)
BEC (Business Email Compromise)	Imitación del estilo del jefe (longitud de frases, vocabulario, uso de mayúsculas, puntuación)	Estilometría (estudio cuantitativo del estilo de escritura)
Ingeniería social en chats	Lenguaje persuasivo, preguntas excesivas, cambios bruscos de tono	Clasificación de texto + detección de anomalías

Notas de ransomware	Frases típicas (“tus archivos están encriptados”, amenazas en mal inglés)	Reconocimiento de entidades + topic modeling
Amenazas generadas por IA (2025-2026)	Texto “demasiado perfecto”, falta de huellas humanas (repetición sutil de patrones, distribución de palabras función)	Modelos estilométricos específicos para distinguir humano vs LLM

El código en Ilustración 14 extrae features (características) lingüísticas útiles para detectar phishing. Usa “textstat” para métricas de legibilidad y el modelo mediano de procesamiento de lenguaje “es_core_news_md” para mayor precisión.

```
import spacy
from collections import Counter

!pip install textstat #Para estadísticas del texto
import textstat

!python -m spacy download es_core_news_md
nlp = spacy.load("es_core_news_md") #Modelo con mayor precisión

def analizar_patrones_phishing(texto):
    doc = nlp(texto.lower())

    # Features básicas
    tokens = [token.lemma_ for token in doc if not token.is_punct and not token.is_stop]
    palabras_urgencia = ["urgente", "inmediato", "ahora", "suspendida", "bloqueada", "verificar", "clic", "acceso"]
    score_urgencia = sum(1 for token in tokens if token in palabras_urgencia)

    # Métricas estilométricas
    oraciones = list(doc.sents)
    long_media_oracion = sum(len(sent) for sent in oraciones) / len(oraciones) if oraciones else 0
    legibilidad = textstat.flesch_reading_ease(texto)  # valores bajos = texto más complejo/difícil

    print(f"Score urgencia: {score_urgencia}")
    print(f"Longitud media oración: {long_media_oracion:.1f}")
    print(f"Legibilidad (Flesch): {legibilidad:.1f} (bajo = sospechoso)")
    print(f"Palabras más usadas: {Counter(tokens).most_common(10)}")

    return score_urgencia > 3 or long_media_oracion < 12 or legibilidad < 40

# Uso
email_sospechoso = "Estimado cliente, su cuenta ha sido suspendida inmediatamente. "
email_sospechoso+="Haga clic aquí para verificar sus datos de forma urgente o perderá el acceso."
analizar_patrones_phishing(email_sospechoso)
```

Ilustración 10. Extracción de features lingüísticas (contenido de autor Rafael Melgarejo)

El resultado es el siguiente:

- Score urgencia: 4
- Longitud media oración: 12.5
- Legibilidad (Flesch): 41.9 (bajo = sospechoso)
- Palabras más usadas: [('estimado', 1), ('cliente', 1), ('suspender', 1), ('inmediatamente', 1), ('hacer', 1), ('clic', 1), ('verificar', 1), ('dato', 1), ('forma', 1), ('urgente', 1)]
- True

Vectorización

Los patrones en el modelo ya no son frases, sino vectores de la forma: [urgente=1, verificar=1, cuenta=1, error=0, vpn=0]. El lenguaje deja de ser texto y pasa a ser geometría matemática. En este caso, el aprendizaje automático descubre: combinaciones frecuentes, correlaciones lingüísticas, y señales predictivas.

Educación de calidad

REFERENCIAS

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