PUCE Virtual

CLASE

6

Procesamiento de Lenguaje Natural

Análisis semántico e intención maliciosa

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 este punto del curso, los estudiantes han aprendido a procesar texto, representarlo computacionalmente y construir modelos capaces de clasificar mensajes según su tipo de amenaza. Sin embargo, muchos ataques reales no utilizan palabras clave evidentes ni patrones superficiales fáciles de detectar. En contextos de ciberseguridad, los atacantes suelen emplear lenguaje ambiguo, indirecto o cuidadosamente diseñado para evadir filtros automáticos. Por esta razón, la Semana 6 introduce un cambio fundamental de enfoque: pasar de la detección basada en palabras a la comprensión del significado del lenguaje, es decir, al análisis semántico.

El análisis semántico permite identificar no solo qué palabras aparecen en un texto, sino qué intención comunica el mensaje. En este sentido, el curso profundiza en conceptos como la intención comunicativa, la pragmática del lenguaje y las señales discursivas de manipulación, elementos clave en ataques de ingeniería social. Además, se introducen técnicas avanzadas como los embeddings semánticos, que permiten representar el significado del texto en espacios vectoriales, y métodos para detectar anomalías lingüísticas que pueden indicar comportamientos maliciosos. Finalmente, se explora el uso de heurísticas híbridas, que combinan reglas lingüísticas y modelos de aprendizaje automático para mejorar la detección de amenazas. De esta manera, la Semana 6 marca la transición hacia sistemas de PLN más sofisticados, capaces de identificar ataques incluso cuando estos no siguen patrones explícitos.

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 3

Análisis semántico e intención maliciosa.

Palabras clave vs. semántica.

Limitaciones de la detección basada en palabras clave

En las primeras etapas del análisis de texto, muchos sistemas de detección de amenazas utilizan enfoques basados en palabras clave. Este método consiste en identificar términos específicos que suelen aparecer en mensajes maliciosos, como:

urgente, verifique, contraseña, suspensión, acceso, confirme

Este enfoque es simple y computacionalmente eficiente, y puede ser útil para detectar patrones evidentes de ataques como el phishing. Sin embargo, presenta una limitación crítica: depende de la presencia explícita de ciertas palabras.

En contextos reales de ciberseguridad, los atacantes suelen adaptar su lenguaje para evitar estos filtros. Por ejemplo:

- “Revise su acceso a la plataforma” en lugar de “verifique su cuenta”.
- “Se recomienda actualizar sus datos” en lugar de “actualice sus credenciales”.

Aunque el significado es similar, las palabras clave pueden no coincidir, lo que reduce la capacidad de detección del sistema.

Hacia el análisis semántico

Para superar estas limitaciones, es necesario avanzar hacia el análisis semántico, que busca comprender el significado del texto más allá de las palabras individuales.

El enfoque semántico permite reconocer que diferentes expresiones pueden transmitir la misma intención, como por ejemplo:

- "verifique su cuenta"
- "confirme su acceso"
- "valide sus credenciales"

Aunque utilizan palabras distintas, todas expresan una misma acción solicitada al usuario, lo cual puede ser indicativo de un intento de manipulación.

Diferencia conceptual (Einsenstein,2019)

Enfoque	Característica principal	limitación
PALABRAS CLAVE	Busca términos específicos	Fácil de evadir
SEMÁTICA	Analiza significados del mensaje	Mayor complejidad

Ejemplo comparativo: supongamos el siguiente texto: “Por favor valide su acceso a la plataforma corporativa”. Entonces:

- Al aplicar el método basado en palabras clave, el sistema busca únicamente ciertas palabras como “verifique”, como no la encuentra, no detecta la amenaza.
- Con el método semántico, el sistema detecta posible intento de ingeniería social, porque interpreta:
  - acción solicitada: validación de acceso
  - contexto: cuenta o sistema
  - intención: interacción del usuario

Implicaciones en ciberseguridad

El uso exclusivo de palabras clave puede generar dos problemas:

- Falsos negativos: ataques que no contienen términos explícitos
- Falsos positivos: mensajes legítimos con palabras “sospechosas”

El análisis semántico reduce estos problemas al considerar el contexto y la intención del mensaje.

Implementación básica en Python

En ilustración 1: sistema basado en palabras clave. Resultado: “False”.

```python
def detectar_keyword(texto):
    keywords = ["urgente", "verifique", "contraseña"]
    return any(palabra in texto.lower() for palabra in keywords)

print(detectar_keyword("valide su acceso ahora"))
```

Ilustración 1. Python: Sistema basado en palabras clave (creación de autor Rafael Melgarejo)

En ilustración 2: enfoque con vectorización semántica. El resultado “[[0.20199]]”, indicando que los textos tienen un nivel de similaridad semántica, aunque no compartan las mismas palabras. Es decir, las palabras pueden cambiar pero la intención permanece.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

textos = [
    "verifique su cuenta",
    "valide su acceso",
]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(textos)

similitud = cosine_similarity(X[0], X[1])
print(similitud)
```

Ilustración 2. Vectorización Semántica (creación de autor Rafael Melgarejo)

Intención comunicativa y pragmática.

En el análisis semántico no basta con identificar el significado literal de las palabras; es necesario comprender qué intención tiene el emisor del mensaje. Este enfoque corresponde al estudio de la intención comunicativa, que busca responder:

¿Qué acción intenta provocar este mensaje en el receptor?

En ciberseguridad, esta pregunta es fundamental, ya que muchos ataques no dependen únicamente del contenido textual, sino del efecto que buscan generar en el usuario.

Pragmática del lenguaje

Es la rama de la lingüística que estudia cómo el contexto influye en la interpretación del lenguaje. En otras palabras, analiza cómo se usa el lenguaje en situaciones reales. Esto implica que el significado de un mensaje no depende solo de las palabras, sino también de:

- el contexto en el que se emite,
- la relación entre emisor y receptor,
- la intención detrás del mensaje.

Para conocer más sobre la intención comunicativa del texto revise el video del profesor Astorga “La INTENCIÓN COMUNICATIVA DE UN TEXTO explicada con ejemplos” en: https://www.youtube.com/watch?v=TJv4-p9LwVc

Tipos de intención en ciberseguridad

El análisis semántico permite ubicar un mensaje dentro del espectro de intenciones (Jurafsky, 2023).

tipo de intención	ejemplo	riesgo
Informativa	El sistema estará en mantenimiento	Bajo
legítimo	Ingrese al sistema para revisar su cuenta	Medio
persuasivo	Debe actualizar sus datos ahora	Alto
manipulativo	Si no actúa, su cuenta será suspendida	Crítico

Actos de habla

Desde la teoría lingüística, los mensajes pueden clasificarse como actos de habla, es decir, acciones realizadas a través del lenguaje. En ciberseguridad, los más relevantes son:

- Directivos → buscan que el usuario haga algo
- Comisivos → prometen una acción (ej. recompensa)
- Asertivos → presentan información como verdadera

Implicaciones para el análisis automático.

Un sistema basado solo en palabras puede no detectar diferencias entre ambos textos. Sin embargo, un sistema que incorpora análisis pragmático puede identificar:

- nivel de urgencia
- presencia de coerción
- tipo de acción solicitada

Esto permite detectar intenciones maliciosas incluso cuando el lenguaje es sutil.

Representación computacional de la intención

En PLN, la intención puede modelarse mediante:

- patrones lingüísticos (ej. verbos imperativos),
- estructuras sintácticas,
- embeddings semánticos,
- modelos de clasificación de intención.

Una lógica como de ilustración 3, puede combinarse con modelos de ML.

```
if contiene_urgencia and contiene_accion:
    posible_ataque = True
```

Ilustración 3. Lógica para la intención (creación de autor Rafael Melgarejo)

Señales de ingeniería social.

Las señales de ingeniería social son patrones lingüísticos y discursivos diseñados para manipular el comportamiento del usuario, explotando sesgos cognitivos como el miedo, la urgencia, la autoridad o la confianza. A diferencia de los ataques

puramente técnicos, la ingeniería social se basa en el uso estratégico del lenguaje para inducir decisiones rápidas o poco reflexivas.

En términos de PLN, estas señales no siempre se detectan por palabras específicas, sino por combinaciones de intención, tono y contexto, lo que las convierte en un problema ideal para el análisis semántico.

Principales categorías de señales

- Urgencia artificial: Mensajes que presionan al usuario a actuar rápidamente, reduciendo su capacidad de análisis crítico.
- Suplantación de autoridad: El atacante se hace pasar por una entidad legítima (banco, empresa, administrador del sistema).
- Persuasión emocional o coercitiva: Se apela a emociones como miedo, recompensa o culpa.

Implementación Python

Ilustración 4 ejemplifica una detección automática de señales. Mientras el score sea más alto, hay mayor probabilidad de ataque y viceversa.

```python
def detectar_senales(texto):
    urgencia = ["urgente", "ahora", "inmediatamente"]
    autoridad = ["seguridad", "banco", "administrador"]
    accion = ["verificar", "actualizar", "confirmar"]
    consecuencia = ["bloqueo", "suspensión", "acceso restringido"]

    score = 0
    texto_lower = texto.lower()

    # Itera a través de cada palabra clave en cada lista y cuenta todas sus ocurrencias
    for keyword in urgencia:
        score += texto_lower.count(keyword)
    for keyword in autoridad:
        score += texto_lower.count(keyword)
    for keyword in accion:
        score += texto_lower.count(keyword)
    for keyword in consecuencia:
        score += texto_lower.count(keyword)

    return score

texto = "Debe verificar su cuenta inmediatamente"
print(f"Texto: '{texto}'")
print(f"Score de la señal: {detectar_senales(texto)}")
```

Ilustración 4. Detección automática de señales (creación de autor Rafael Melgarejo)

Embeddings y similitud semántica.

Embedding

Es una representación vectorial del lenguaje que permite capturar el significado semántico de palabras, frases o textos completos en un espacio matemático. A diferencia de representaciones como BoW o TF-IDF (que se basan en frecuencia de palabras), los embeddings buscan representar relaciones de significado. En términos simples:

palabra → vector numérico → significado en el espacio semántico

Esto permite que palabras o frases con significados similares estén cercanas entre sí en el espacio vectorial, incluso si no comparten términos exactos.

Tipos de Embeddings

Según el nivel de complejidad (Jurafsky,2023). Para ciberseguridad, los más útiles suelen ser los sentence embeddings, ya que permiten analizar mensajes completos. En ciberseguridad los embeddings permiten:

- detectar ataques sin palabras clave explícitas
- identificar patrones de lenguaje similares
- agrupar mensajes sospechosos
- mejorar clasificadores de texto

tipo	característica
Word embeddings	Representan palabras individuales (Word2Vec, GloVe)
Sentence embeddings	Representan frases completas
Contextual embeddings	Consideran el contexto (BERT, transformers)

Los embeddings tienen limitaciones como:

- requieren mayor capacidad computacional
- pueden ser difíciles de interpretar
- necesitan datasets representativos

Similitud semántica

Una vez que los textos están representados como vectores, se puede medir su similitud mediante métricas como la similitud del coseno:

- similitud ≈ 1 → textos muy similares
- similitud ≈ 0 → textos diferentes

Esto permite detectar:

- variantes de ataques
- reformulaciones del mismo mensaje
- intentos de evasión

Para conocer más sobre la similitud semántica, revise el video de CodelyTY "Qué son las BÚSQUEDAS SEMÁNTICAS y los EMBEDDINGS: IA con tu base de datos" en https://www.youtube.com/watch?v=5rvUTeb0be4

Ejemplo práctico en Python

La interpretación del texto ejemplo en el código de Ilustración 5 es:

- alta similitud entre textos maliciosos
- baja similitud entre texto malicioso y legítimo

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

textos = [
    "verifique su cuenta",
    "valide su acceso",
    "reunión del equipo mañana"
]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(textos)

sim_1_2 = cosine_similarity(X[0], X[1])
sim_1_3 = cosine_similarity(X[0], X[2])

print("Similitud (ataque vs ataque):", sim_1_2)
print("Similitud (ataque vs legítimo):", sim_1_3)
```

```
Similitud (ataque vs ataque): [[0.224325]]
Similitud (ataque vs legítimo): [[0.]]
```

Ilustración 5. Implementación de Vectorización y Similaridad de Coseno (creación de autor Rafael Melgarejo)

Detección de anomalías lingüísticas.

Una anomalía lingüística es una desviación respecto al uso normal del lenguaje dentro de un contexto específico. En ciberseguridad, estas anomalías pueden indicar que un mensaje no sigue los patrones habituales de comunicación de una organización, lo cual puede ser una señal de actividad maliciosa.

A diferencia de la detección por palabras clave o incluso por semántica directa, este enfoque se basa en identificar lo inusual, es decir:

¿Qué tiene este mensaje que no encaja con el comportamiento normal?

Tipos de anomalías en ciberseguridad

- Léxicas: uso de palabras, relacionado con: vocabulario inusual, términos fuera del dominio organizacional, combinaciones poco frecuentes.
- Sintácticas (estructura): construcciones gramaticales atípicas, errores de redacción, estructuras no habituales.
- Discursivas (tono e intención): cambios abruptos de tono mezcla de formalidad e informalidad, urgencia inesperada.
- Contextuales: mensajes fuera de horario, solicitudes inusuales par el rol del usuario, cambios en patrones de comunicación.

Enfoques de detección

Se listan algunos enfoques. Ilustración 6 muestra implementación simple en Python.

- Basado en reglas: se definen patrones de comportamiento sospechoso.

- Basado en estadísticas: se analizan distribuciones del lenguajes: frecuencia de palabras, longitud de mensajes, patrones de uso.
- Basado en embeddings: se compara el mensaje con el lenguaje típico.
- Modelos de detección de Anomalías:
    - Isolation Forest
    - One-Class SVM
    - Autoencoders

```python
#Ejemplo de Detección de anomalías basado en reglas
if "inmediatamente" in texto and "acceso" in texto:
    alerta = True

#Ejemplo Detección de anomalías basado en estadísticas
longitud_media = df["texto"].str.len().mean()

if len(texto) > longitud_media * 2:
    print("Mensaje anómalo")

#Ejemplo Detección de anomalías basado en embeddings
if similitud < umbral:
    posible_anomalia = True
```

Ilustración 6. Ejemplos de Detección de Anomalías (creación de autor Rafael Melgarejo)

Relación con reto 3

El reto 2 clasifica amenazas conocidas, el reto 3 detecta patrones no evidentes (amenazas desconocidas). Como apoyo al reto 3, ilustración 7 tiene un ejemplo práctico en Python:

```python
textos_normales = [
    "reunion del equipo mañana",
    "actualizacion del sistema programada",
    "ticket resuelto correctamente"
]

nuevo_texto = "debe verificar su cuenta inmediatamente"

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(textos_normales + [nuevo_texto])

similitudes = cosine_similarity(X[-1], X[:-1])

print("Similitud con mensajes normales:", similitudes)

if similitudes.max() < 0.2:
    print("Posible anomalía detectada")
```

```
Similitud con mensajes normales: [[0. 0. 0.]]
Posible anomalía detectada
```

Ilustración 7. Ejemplo práctico en Python (creación de autor Rafael Melgarejo)

Modelado de intención e heurísticas híbridas.

Consiste en identificar automáticamente qué quiere lograr un mensaje, es decir, clasificar la intención comunicativa del texto (informar, solicitar, persuadir, manipular, etc.). En ciberseguridad, este proceso es clave porque muchos ataques no se distinguen por su forma, sino por su propósito. El objetivo del modelado de intención es transformar este análisis en un proceso automatizable mediante PLN.

Tipos de intención relevantes en ciberseguridad (Eisenstein,2019)

Un sistema puede clasificar mensajes según su intención:

INTENCIÓN	DESCRIPCIÓN	NIVEL DE RIESGO
Informativa	Comunica un hecho	Bajo
Solicitud	Pide una acción	Medio
Persuasiva	Intenta convencer	Alto
Coercitiva	Presiona o amenaza	Crítico

Este tipo de clasificación permite priorizar alertas en un SOC

¿Por qué no basta un solo enfoque?

Ningún método por sí solo es suficiente:

- Palabras clave → fácilmente evadibles
- Modelos ML → requieren muchos datos
- Reglas → poco flexibles

Por ello, se utilizan heurísticas híbridas que combinan:

reglas lingüísticas + modelos de aprendizaje automático + análisis semántico

Heurísticas híbridas

Las heurísticas híbridas son estrategias que integran múltiples fuentes de información para mejorar la detección. Ejemplo de combinación:

- **embeddings** → capturan significado
- **reglas** → detectan patrones específicos
- **métricas** → evalúan riesgo

Esto permite construir sistemas más robustos frente a ataques reales. Ejemplo: en el mensaje: “"El equipo de seguridad solicita validar su cuenta inmediatamente". El sistema puede analizar:

- **autoridad** → "equipo de seguridad"
- **acción** → "validar"
- **urgencia** → "inmediatamente"

→ resultado: alta probabilidad de intención maliciosa

Ilustración 8 presenta una implementación simple de heurística híbrida:

```python
def evaluar_intencion(texto):
    texto = texto.lower()

    urgencia = ["urgente", "inmediatamente", "ahora"]
    autoridad = ["seguridad", "banco", "administrador"]
    accion = ["verifique", "actualice", "confirme", "valide"]

    score = 0

    if any(p in texto for p in urgencia):
        score += 1
    if any(p in texto for p in autoridad):
        score += 1
    if any(p in texto for p in accion):
        score += 1

    if score >= 2:
        return "posible ataque"
    else:
        return "probablemente legítimo"


texto = "Debe validar su cuenta inmediatamente"
print(evaluar_intencion(texto))
```

```
probablemente legítimo
```

Ilustración 8. Heurística híbrida (creación de autor Rafael Melgarejo)

Extensión con embeddings

Se puede combinar con similitud semántica:

```
if similitud_con_ataques > umbral:
    score += 1
```

Esto permite detectar:

- variantes de ataques
- lenguaje evasivo
- mensajes reformulados

Ventajas y desventajas del enfoque híbrido

El objetivo del enfoque híbrido es detectar ataques no es solo clasificar textos, es entender la intención detrás del lenguaje. Este enfoque tiene sus ventajas y desventajas:

- Ventajas
  - mayor robustez ante evasión
  - mejor interpretación del lenguaje
  - adaptable a diferentes contextos
  - útil con pocos datos
- Desventajas
  - requiere diseño cuidadoso de reglas
  - puede ser difícil de mantener

- necesita calibración de umbrales

Articulación del contenido con los retos:

La integración de los tres primeros retos del curso responde a una lógica progresiva de construcción de un sistema de PLN aplicado a ciberseguridad, donde cada reto desarrolla un componente fundamental que luego se articula en una solución más compleja.

- En el Reto 1, se establece la base del sistema mediante la recolección, limpieza y comprensión del texto. Aquí el estudiante transforma datos no estructurados (correos, logs, reportes) en un dataset preparado y analizable, identificando patrones lingüísticos iniciales. Este paso es crítico, ya que la calidad de los datos condiciona el desempeño de cualquier modelo posterior.
- En el Reto 2, sobre esta base, se construye el primer componente funcional: un clasificador automático de amenazas. El estudiante aplica técnicas de vectorización (como TF-IDF o embeddings) y entrena modelos capaces de distinguir entre distintos tipos de mensajes (phishing, legítimos, etc.). Este reto introduce la capacidad de detección automática basada en patrones aprendidos.
- Al momento, el Reto 3 amplía el sistema hacia un nivel más avanzado mediante el análisis semántico y la detección de intención maliciosa. Aquí se incorporan técnicas para identificar mensajes evasivos, ambiguos o reformulados, utilizando embeddings, detección de anomalías y heurísticas híbridas. Este componente permite superar las limitaciones de los modelos basados únicamente en clasificación.

En conjunto, los tres retos se integran como un pipeline:

Datos → Preparación → Clasificación → Análisis semántico

donde cada etapa aporta una capa de inteligencia. Esta integración constituye la base directa para el Reto 4, en el que todos estos componentes se consolidan en un sistema completo de alerta temprana para ciberseguridad.

Glosario:

- **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