INTELIGENCIA ARTIFICIAL: PROCESAMIENTO DEL LENGUAJE NATURAL CON SOFTWARE LIBRE

JUAN PABLO LUNA FELIPEZ, Ph.D.

jplunaf@gmail.com

Ingeniería Informática

Universidad Nacional "Siglo XX"

Llallagua, Bolivia

RESUMEN

El Procesamiento del Lenguaje Natural (PLN) es una rama de la inteligencia artificial que permite a las máquinas entender, interpretar y generar lenguaje humano. Entre sus aplicaciones destacan: análisis de sentimientos, clasificación de textos, traducción automática, resumen de textos, reconocimiento de voz y texto, y detección de entidades como nombres o lugares. El PLN se apoya en herramientas y bibliotecas de software libre, especialmente en lenguajes como Python, R, Julia, Java y C++, y librerías como spaCy, NLTK, Transformers, Gensim y TextBlob. El proceso incluye técnicas como tokenización, eliminación de palabras vacías (stopwords), stemming, lematización, POS tagging y word embeddings.

1. INTRODUCCIÓN

El lenguaje es una de las herramientas más poderosas del ser humano para comunicar pensamientos, emociones e información. En la era digital, la necesidad de que las máquinas comprendan y generen lenguaje humano dieron origen a una disciplina clave dentro de la inteligencia artificial: el Procesamiento del Lenguaje Natural (PLN). Esta área busca desarrollar sistemas capaces de interpretar, analizar y producir lenguaje de manera automática, facilitando la interacción entre humanos y computadoras.

Gracias al uso de software libre, hoy en día es posible acceder a poderosas herramientas y bibliotecas que permiten implementar soluciones de PLN de forma accesible y eficiente. Tecnologías como el análisis de sentimientos, la clasificación automática de textos o los asistentes virtuales son solo algunos ejemplos del impacto real que tiene esta disciplina en campos como el marketing, la salud, la educación y el servicio al cliente. Este trabajo explora los fundamentos del PLN, sus principales tareas y aplicaciones, así como las herramientas de código abierto más utilizadas para su desarrollo.

2. DESARROLLO

Procesamiento del lenguaje natural (PLN)

Disciplina de la IA, busca que las computadoras comprendan y procesen el lenguaje humano de forma natural y eficiente. Puede o no, usar aprendizaje automático

Tareas

- **Comprensión del lenguaje natural (NLU):** implica comprender e interpretar el lenguaje humano,
- como el reconocimiento de voz, la clasificación de texto, el análisis de sentimientos y la extracción de información.
- **Generación de lenguaje natural (NLG):** implica generar texto legible por humanos a partir de datos estructurados, como resúmenes de texto, sistemas de diálogo y traducción de idiomas.

Aplicaciones de NLP:

- **Análisis de Sentimientos:** Detectar emociones o intenciones en textos. Por ejemplo, opiniones en redes sociales, encuestas de satisfacción.
- **Clasificación de Texto:** Categorizar documentos automáticamente. Por ejemplo, filtros de spam, categorización de correos o noticias.
- **Reconocimiento de Entidades (NER):** Identificar nombres de personas, lugares, fechas, organizaciones. Por ejemplo, chatbots, sistemas de preguntas y respuestas.
- **Traducción automática:** Convertir texto de un idioma a otro. Por ejemplo, Google Translate, DeepL.

- **Resumen Automático:** Reducir textos largos conservando ideas principales. Por ejemplo, noticias, documentos legales, artículos científicos.
- **Generación de Texto:** Crear texto coherente de manera automática. Por ejemplo, chatGPT, generación de noticias automáticas.
- **Análisis Morfosintáctico (POS tagging):** Identificar la función gramatical de cada palabra. Por ejemplo, preprocesamiento para traductores y chatbots.
- **Extracción de Información:** Detectar datos clave en grandes volúmenes de texto. Por ejemplo, minería de datos, informes automáticos.
- **Reconocimiento de Voz a Texto (ASR):** Convertir voz en texto escrito. Por ejemplo, asistentes virtuales como Siri, Alexa, Google Assistant.
- **Conversión de Texto a Voz (TTS):** Convertir texto escrito en audio. Por ejemplo, lectores de pantalla, asistentes virtuales.
- **Detección de Idioma:** Identificar en qué idioma está escrito un texto. Por ejemplo, plataformas multilingües, filtros automáticos.
- **Corrección Gramatical:** Detectar y corregir errores en el texto. Por ejemplo, Grammarly, correctores de texto automáticos.

Lenguajes de software libre:

- **Python:** Python Software Foundation License (Libre). Razones para usarse en Enorme ecosistema: NLTK, spaCy, Hugging Face Transformers, **Gensim.** Muy usado en IA y Deep Learning.
- **R:** GPL (Libre). Enfoque estadístico, ideal para análisis de texto académico. Paquetes como tm, quanteda, text.
- **Julia:** MIT (Libre). Alta velocidad, sintaxis moderna, librería TextAnalysis.jl para NLP. Excelente para modelos grandes.
- **Java:** GPL (Libre). Librerías como Apache OpenNLP y Stanford CoreNLP. Muy usado en entornos empresariales y producción.
- **C++ :** GPL / BSD / MIT (según proyecto). Máximo rendimiento. Usado en motores de NLP como spaCy (núcleo) y software que exige velocidad extrema.

Bibliotecas libres Python:

- **NLTK:** Apache 2.0 (Libre). Herramientas clásicas para análisis léxico, tokenización, stemming, POS tagging.
- **spaCy:** MIT (Libre) Procesamiento eficiente de texto, modelos preentrenados, entidades, dependencias.
- **Transformers (Hugging Face):** Apache 2.0 (Libre). Modelos preentrenados: BERT, GPT, RoBERTa, T5 para tareas modernas de NLP.
- **Gensim:** LGPL (Libre). Modelado de tópicos y vectores: Word2Vec, Doc2Vec, FastText.
- **TextBlob:** MIT (Libre). Simplificación de tareas NLP: traducción, análisis de sentimientos, POS tagging.
- **Flair:** MIT (Libre). NLP de última generación con embeddings contextuales, desarrollado por Zalando.
- **Stanza:** Apache 2.0 (Libre). Toolkit de Stanford para NLP: análisis sintáctico, NER, modelos multilingües.
- **Polyglot:** GPLv3 (Libre). Procesamiento multilingüe: detección de idioma, NER, embeddings.

Tokenización: Dividir un texto en partes pequeñas, como palabras o frases.

```python
from nltk.tokenize import word_tokenize
texto = "Hola mundo"
tokens = word_tokenize(texto)
print(tokens)  # ['Hola', 'mundo']
```

Figura 1: Ejemplo práctico, separar la frase "Hola mundo" en las palabras "Hola" y "mundo".

Eliminación de stopwords: Quitar palabras comunes que no aportan mucho significado, como "el", "y", "de".

```python
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
nltk.download('stopwords')

texto = "Este es un ejemplo simple"
tokens = word_tokenize(texto.lower())
stop_words = set(stopwords.words('spanish'))
filtrado = [t for t in tokens if t not in stop_words]
print(filtrado)  # ['ejemplo', 'simple']
```

Figura 2: Ejemplo práctico, en la frase "Este es un ejemplo simple", eliminar palabras como "es", "un".

Stemming: Reducir palabras a su raíz para agrupar variantes.

```python
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
print(stemmer.stem("running"))  # run
```

Figura 3: Ejemplo práctico "corriendo", "corrió" y "correr" se reducen a "corr".

Lematización: Convertir palabras a su forma base real según el contexto.

```python
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("running", pos='v'))  # run
```

Figura 4: Ejemplo práctico, "corriendo" se convierte en "correr".

Etiquetado gramatical (POS Tagging): Identificar la función gramatical de cada palabra (sustantivo, verbo, adjetivo, etc.).

```
from nltk import pos_tag, word_tokenize
tokens = word_tokenize("El gato corre")
tags = pos_tag(tokens)
print(tags)  # [('El', 'DT'), ('gato', 'NN'), ('corre', 'VBZ')]
```

Figura 5: Ejemplo práctico: En "El gato corre", "gato" es sustantivo y "corre" es verbo.

Reconocimiento de entidades nombradas (NER): Detectar nombres propios como personas, lugares u organizaciones en un texto.

```
from nltk import ne_chunk, pos_tag, word_tokenize
import nltk
nltk.download('maxent_ne_chunker')
nltk.download('words')

texto = "Barack Obama nació en Hawaii"
tokens = word_tokenize(texto)
tags = pos_tag(tokens)
tree = ne_chunk(tags)
print(tree)
```

Figura 6: Ejemplo práctico, en "Barack Obama nació en Hawaii", reconocer "Barack Obama" como persona y "Hawaii" como lugar.

N-gramas: Secuencias de “n” palabras juntas para analizar contexto.

```
from nltk.util import ngrams
tokens = ["software", "libre"]
bigrams = list(ngrams(tokens, 2))
print(bigrams)  # [('software', 'libre')]
```

Figura 7: Ejemplo práctico, en "software libre", el bigrama es ("software", "libre").

Conteo de frecuencia: Contar cuántas veces aparece cada palabra en un texto.

```
from collections import Counter
tokens = ["libre", "libre", "software"]
freq = Counter(tokens)
print(freq)  # {'libre': 2, 'software': 1}
```

Figura 8: Ejemplo práctico: En "libre libre software", "libre" aparece 2 veces.

Clasificación de texto: Asignar una categoría a un texto, como spam o no spam.

```
texto = "Gana dinero rápido"
print("spam" if "dinero" in texto else "ham")  # spam
```

Figura 9: Ejemplo práctico, detectar si un correo es spam según palabras clave.

Word Embeddings: Representar palabras con vectores numéricos que capturan su significado.

```
from gensim.models import Word2Vec

sentences = [["software", "libre"], ["libertad", "comunidad"]]
model = Word2Vec(sentences, vector_size=10, min_count=1)
print(model.wv['software'])  # vector numérico
```

Figura 10: Ejemplo práctico: Palabras similares tienen vectores cercanos, como "rey" y "reina".

Flujo típico de NLP:

Preprocesamiento de texto: Preparar el texto para su análisis: limpieza, normalización y tokenización.

```
# Limpieza y normalización
texto = "¡Hola Mundo! ¿Cómo estás?"
texto_limpio = texto.lower().replace("¡", "").replace("!", "").replace("¿",
"").replace("?", "")
print(texto_limpio)
# Salida: hola mundo cómo estás

# Tokenización
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')

tokens = word_tokenize(texto_limpio)
print(tokens)
# Salida: ['hola', 'mundo', 'cómo', 'estás']
```

Figura 11: Preparación del texto para analizarlo

Eliminación de stopwords: Quitar palabras sin significado relevante.

```
from nltk.corpus import stopwords
nltk.download('stopwords')

stop_words = set(stopwords.words('spanish'))
tokens_filtrados = [t for t in tokens if t not in stop_words]
print(tokens_filtrados)
# Ejemplo salida: ['hola', 'mundo']
```

Figura 12: Eliminación de stopwords

Stemming y Lematización: Reducir palabras a su raíz o forma base para agrupar variantes.

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
import nltk
nltk.download('wordnet')

stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

print(stemmer.stem("running"))        # run
print(lemmatizer.lemmatize("running", pos='v'))  # run
```

Figura 13: Stemming y lematización

- **Etiquetado gramatical (POS Tagging):** Asignar categorías gramaticales a cada palabra.

```
from nltk import pos_tag

nltk.download('averaged_perceptron_tagger')

tags = pos_tag(tokens_filtrados)
print(tags)
# Ejemplo salida: [('hola', 'NN'), ('mundo', 'NN')]
```

Figura 14: Etiquetado gramatical

Reconocimiento de entidades nombradas (NER): Detectar nombres propios como personas, lugares u organizaciones.

```
from nltk import ne_chunk

nltk.download('maxent_ne_chunker')
nltk.download('words')

arbol = ne_chunk(tags)
print(arbol)
```

Figura 15: Reconocimiento de entidades nombradas

Extracción de características: Convertir texto en datos numéricos para modelos (Bag of Words, TF-IDF, embeddings).

```
from collections import Counter

frecuencia = Counter(tokens_filtrados)
print(frecuencia)
# Ejemplo salida: Counter({'hola': 1, 'mundo': 1})
```

Figura 16: Extracción de características

Modelado y aprendizaje automático: Entrenar modelos para tareas como clasificación o análisis de sentimientos.

```
# Clasificación simple basada en palabras clave
texto = "Gana dinero rápido"
print("spam" if "dinero" in texto.lower() else "ham")
# Salida: spam
```

Figura 17: Clasificación simple basada en palabra clave

Análisis de Sentimientos: Determinar si un texto expresa opinión positiva, negativa o neutral.

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()
scores = sid.polarity_scores("I love this product!")
print(scores)
# Ejemplo salida: {'neg': 0.0, 'neu': 0.294, 'pos': 0.706, 'compound': 0.6696}
```

Figura 18: Análisis de sentimientos

Modelado de temas: Identificar temas principales en textos largos o colecciones.

```python
from gensim import corpora, models

documentos = [
    "El fútbol es un deporte popular",
    "La política afecta la economía",
    "El baloncesto es divertido"
]

texts = [doc.lower().split() for doc in documentos]
diccionario = corpora.Dictionary(texts)
corpus = [diccionario.doc2bow(text) for text in texts]

lda = models.LdaModel(corpus, num_topics=2, id2word=diccionario, passes=10)
for idx, topic in lda.print_topics(-1):
    print(f"Tema {idx}: {topic}")
```

Figura 19: Modelado de temas

Comprensión del lenguaje natural (NLU) básica: Interpretar la intención o significado detrás del texto.

```
nltk.download('wordnet')

texto = "¿Puedes reservarme un taxi para mañana?"

# Tokenizar el texto
tokens = word_tokenize(texto.lower())

# Lematizar cada palabra
lemmatizer = WordNetLemmatizer()
lemmas = [lemmatizer.lemmatize(token) for token in tokens]

# Definir palabras clave en su forma base
intencion = ("reservar" in lemmas or "reservar" in tokens) and "taxi" in lemmas

if intencion:
    print("Intento detectado: Reservar un taxi")
else:
    print("Intento desconocido")
```

Figura 20: Interpretación del significado del texto

3. CONCLUSIÓN

El Procesamiento del Lenguaje Natural permite a las máquinas comprender y generar lenguaje humano, facilitando numerosas aplicaciones como asistentes virtuales, análisis de sentimientos y traducción automática. Gracias al software libre, estas tecnologías están al alcance de todos, impulsando la innovación y el aprendizaje. En un entorno cada vez más digital, el PLN se vuelve esencial para aprovechar al máximo la información escrita y mejorar la interacción entre personas y sistemas inteligentes.

SOBRE EL AUTOR

Doctor en Ciencias de la Computación, Máster en Ciencias de la Computación con mención en Seguridad Informática y Software Libre, Máster en Educación Superior, cuenta con Diplomados en Preparación Evaluación y Gestión de Proyectos, Formación Basada por Competencias y Metodología de la Investigación Científica.

Ha sido Director de la Carrera Ingeniería Informática y actualmente es Presidente de la Sociedad Científica de Docentes (SOCID) de la Universidad Nacional “Siglo XX”, Coordinador del Instituto de Investigación y Desarrollo de Aplicaciones Informáticas IIDAI, Director de la Revista Científica “Ciencia y Tecnología Informática”, Director de las publicaciones Memorias SOCID, Memorias Ciencia y Tecnología Informática y Docente Universitario Titular en la carrera Ingeniería Informática de la Universidad Nacional “Siglo XX”, Docente de pre y postgrado, organiza varios eventos académicos.

Expositor y organizador en varios eventos académicos.

Publicó varias Revistas Científicas, Artículos y Libros, recibió varios reconocimientos y premios.

Figura 21: Fotografía de presentación de la ponencia.