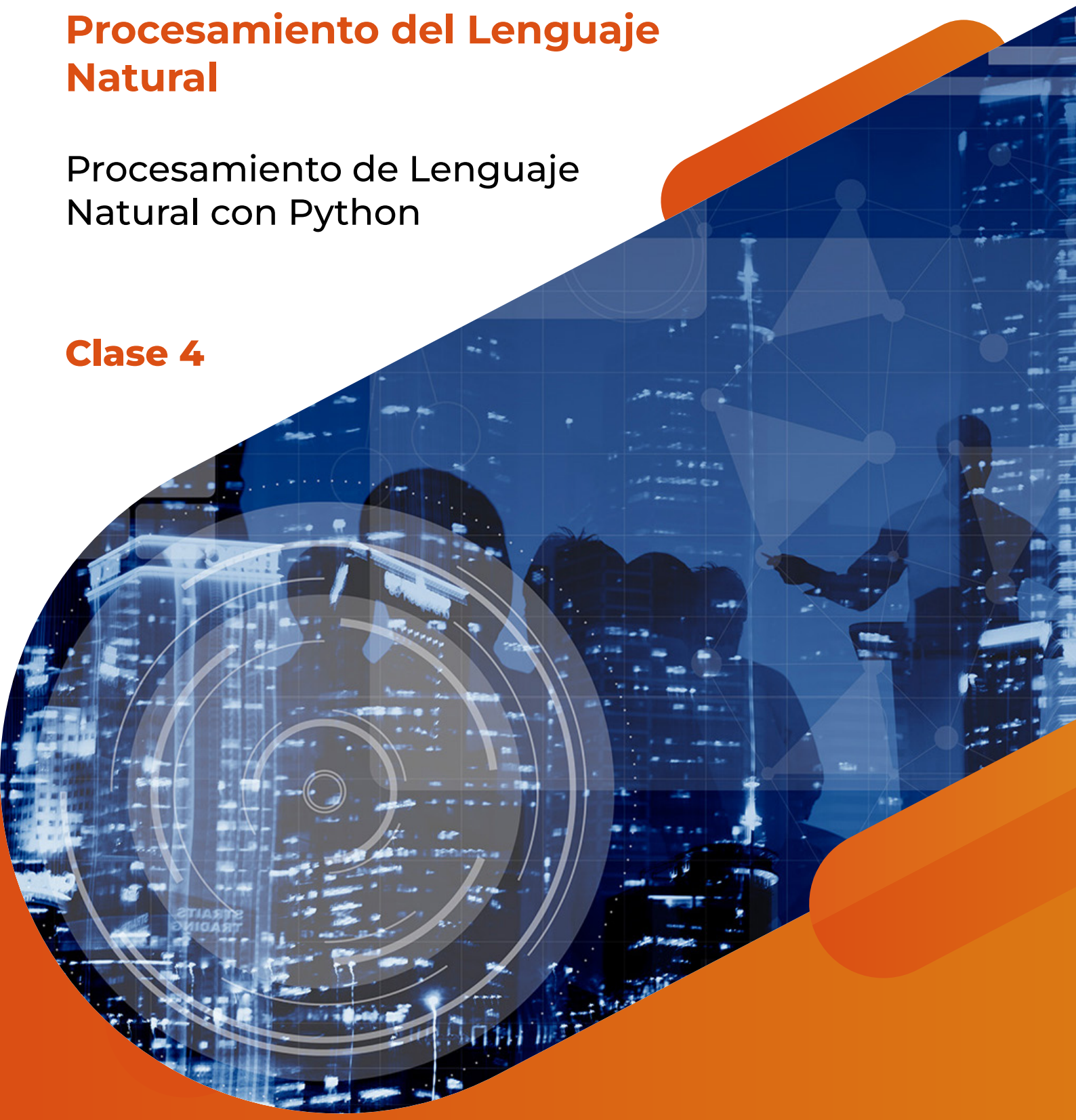
PUCE Virtual
EDUCACIÓN FLEXIBLE

Procesamiento del Lenguaje Natural

Procesamiento de Lenguaje Natural con Python

Clase 4



Maestria en Educacion en Inteligencia Artificial y Entornos Virtuales

La excelencia no se improvisa

Introducción

Esta clase se enfoca en el Procesamiento de Lenguaje Natural (PLN) con Python, presentando tanto sus fundamentos teóricos como su implementación mediante diversas bibliotecas especializadas. Se exploran herramientas como NLTK, que facilita la tokenización y el análisis sintáctico; spaCy, destacada en la extracción de entidades nombradas; TextBlob, útil para la traducción automática; y Gensim, clave para el modelado de temas. Además, se introduce el aprendizaje profundo aplicado al PLN, con énfasis en redes neuronales y modelos de atención y transformadores, esenciales para tareas avanzadas como la comprensión semántica y la generación de texto.

Los resultados de aprendizaje incluyen analizar los conceptos y técnicas de PLN en ambientes educativos. Los estudiantes desarrollarán habilidades en preprocesamiento de datos textuales, extracción de información relevante y aplicación de modelos de inteligencia artificial al lenguaje natural. Al finalizar, serán capaces de implementar soluciones prácticas en PLN, comprendiendo tanto las ventajas como las limitaciones de cada herramienta, lo que les permitirá integrarlas en proyectos reales de análisis y automatización del lenguaje.

Clase 4

7. Procesamiento de Lenguaje Natural con Python

7.1. Introducción

El Procesamiento de Lenguaje Natural (PLN) es una disciplina de la inteligencia artificial que permite a las computadoras entender, interpretar y generar lenguaje humano de manera eficiente. Su aplicación se extiende desde motores de búsqueda hasta asistentes virtuales, traductores automáticos y sistemas de análisis de sentimientos. Python se ha consolidado como el lenguaje de programación predilecto para el PLN debido a su sintaxis clara, su ecosistema de bibliotecas especializadas y su comunidad activa.

Entre las bibliotecas más utilizadas para el PLN en Python (Python, 2025) se encuentran NLTK, Natural Language Toolkit (NLTK, 2024), spaCy (Explosion, 2025) y Transformers (Hugging Face, 2025). NLTK es una biblioteca académica robusta que proporciona herramientas para la tokenización, el etiquetado gramatical y el análisis sintáctico, lo que la hace ideal para la investigación y la enseñanza. Por otro lado, spaCy está diseñado para aplicaciones industriales y se caracteriza por su rapidez y eficiencia en el procesamiento de grandes volúmenes de texto. Finalmente, la biblioteca Transformers de Hugging Face ha revolucionado el PLN mediante el uso de modelos preentrenados basados en arquitecturas de aprendizaje profundo como BERT y GPT, permitiendo resultados de vanguardia en tareas como la generación de texto y el reconocimiento de entidades nombradas.

7.2. NTLK: tokenización

La tokenización es una de las primeras y más fundamentales tareas en el Procesamiento de Lenguaje Natural (PLN), permitiendo dividir el texto en unidades más pequeñas denominadas tokens. En Python, la biblioteca NLTK (Natural Language Toolkit) proporciona herramientas robustas para llevar a cabo esta tarea, facilitando la manipulación y análisis del lenguaje natural en diversas aplicaciones.

NLTK ofrece diferentes métodos de tokenización dependiendo del nivel de granularidad requerido. La tokenización de palabras puede realizarse con nltk.word_tokenize(), que divide un texto en palabras individuales, considerando signos de puntuación y contracciones. En la Figura 1 se puede ver un ejemplo.

Figura 1. Tokenización en palabras con NLTK

```
from nltk.tokenize import word_tokenize
text = "El PLN con Python es poderoso y versátil."
tokens = word_tokenize(text)
print(tokens)

# Se imprimirá:
# ['El', 'PLN', 'con', 'Python', 'es', 'poderoso', 'y', 'versátil', '.']
```

Nota: Creado por Diego Ordoñez (2025)

Por otro lado, nltk.sent_tokenize() permite dividir un texto en oraciones, lo que es útil en tareas de análisis sintáctico y resumen automático. En la Figura 2 puede verse un ejemplo de su aplicación.

Figura 2. Tokenización en oraciones con NLTK

```
from nltk.tokenize import sent_tokenize
text = "El PLN es una rama de la IA. Python es una herramienta clave."
sentences = sent_tokenize(text)
print(sentences)

# Se imprimirá:
# ['El PLN es una rama de la IA.', 'Python es una herramienta clave.']
```

Nota: Creado por Diego Ordoñez (2025)

Gracias a su flexibilidad y potencia, NLTK facilita el preprocesamiento del lenguaje natural, permitiendo que modelos más avanzados extraigan información significativa con mayor precisión.

7.3. spaCy: extracción de NER

El reconocimiento de entidades nombradas (NER, por sus siglas en inglés) es una tarea fundamental en el PLN que permite identificar y clasificar entidades como nombres de personas, organizaciones, ubicaciones y fechas dentro de un texto. En Python, la biblioteca spaCy se destaca por su eficiencia y velocidad en la extracción de NER, lo que la hace ideal para aplicaciones en entornos de producción.

spaCy proporciona modelos preentrenados que facilitan la detección de entidades con gran precisión. La implementación en la Figura 3 ilustra el uso de spaCy para extraer entidades nombradas en un texto: en este ejemplo, spaCy reconoce "Apple" como una organización (ORG), "Madrid" como una ubicación (LOC) y "2025" como una fecha (DATE).

Figura 3. Extracción de NER con spaCy

```python
import spacy

# Cargar modelo en español
nlp = spacy.load("es_core_news_sm")
text = "Apple abrirá una nueva sede en Madrid en 2025."

# Procesar el texto
doc = nlp(text)

# Extraer entidades
for ent in doc.ents:
    print(ent.text, ent.label_)

# Se imprimirá:
# Apple ORG
# Madrid LOC
# 2025 DATE
```

Nota: Creado por Diego Ordoñez (2025)

Además de su capacidad para detectar entidades, spaCy permite mejorar la precisión del modelo mediante entrenamiento personalizado con datos específicos del dominio. Gracias a su rapidez y flexibilidad, spaCy se ha convertido en una herramienta esencial para tareas como el análisis de noticias, la gestión documental y la extracción de información en sectores como la salud y las finanzas.

7.4. TextBlob: Traducción automática

La traducción automática es una de las aplicaciones más relevantes del procesamiento de lenguaje natural (PLN), facilitando la comunicación entre idiomas con rapidez y eficiencia. En Python, TextBlob (Loria, 2025) es una biblioteca que permite realizar traducciones de manera sencilla utilizando la API de Google Translate. Aunque no es tan robusta como otros modelos avanzados basados en redes neuronales, su facilidad de uso y accesibilidad la convierten en una opción popular para aplicaciones generales de traducción.

Funcionamiento de la Traducción con TextBlob

TextBlob proporciona una interfaz intuitiva para traducir textos de un idioma a otro con una sola línea de código. En la Figura 4 se muestra un ejemplo básico.

Figura 4. Traducción con TextBlob

```python
from textblob import TextBlob

text = "El procesamiento de lenguaje natural es una disciplina fascinante."
blob = TextBlob(text)
translated_text = blob.translate(to='en')
print(translated_text)

# Se imprimirá:
# "Natural language processing is a fascinating discipline."
```

Nota: Creado por Diego Ordoñez (2025)

Beneficios y Limitaciones de TextBlob para Traducción

Uno de los principales beneficios de TextBlob es su simplicidad. A diferencia de otras herramientas más complejas, no requiere configuración avanzada, lo que la hace ideal para proyectos pequeños o educativos. Además, aprovecha la API de Google Translate (Google, 2025), lo que le permite acceder a una amplia gama de idiomas y mejorar la precisión con el tiempo.

Sin embargo, TextBlob presenta algunas limitaciones. Primero, al depender de una API externa, requiere conexión a Internet y puede estar sujeto a restricciones de uso o cambios en la disponibilidad del servicio. En segundo lugar, no ofrece el mismo nivel de personalización que modelos avanzados como MarianNMT (Microsoft, 2025) o el servicio de traducción neuronal de DeepL (DeepL, 2025), que permiten adaptar la traducción según el contexto específico o entrenar modelos personalizados.

Aplicaciones de TextBlob en PLN

A pesar de sus limitaciones, TextBlob es una herramienta útil en múltiples escenarios, como la creación de aplicaciones web con traducción automática, el procesamiento de datos multilingües y la asistencia en el aprendizaje de idiomas. En análisis de sentimientos, por ejemplo, la traducción de textos a un idioma específico puede mejorar la precisión de los modelos al trabajar con datasets en varios idiomas.

Aunque TextBlob no es la solución más avanzada en traducción automática, su facilidad de uso y accesibilidad la convierten en una opción viable para tareas básicas y proyectos que requieren traducción rápida sin una infraestructura compleja. Para aplicaciones más exigentes, combinar TextBlob con modelos neuronales avanzados puede ser una estrategia efectiva para mejorar la calidad y precisión de las traducciones.

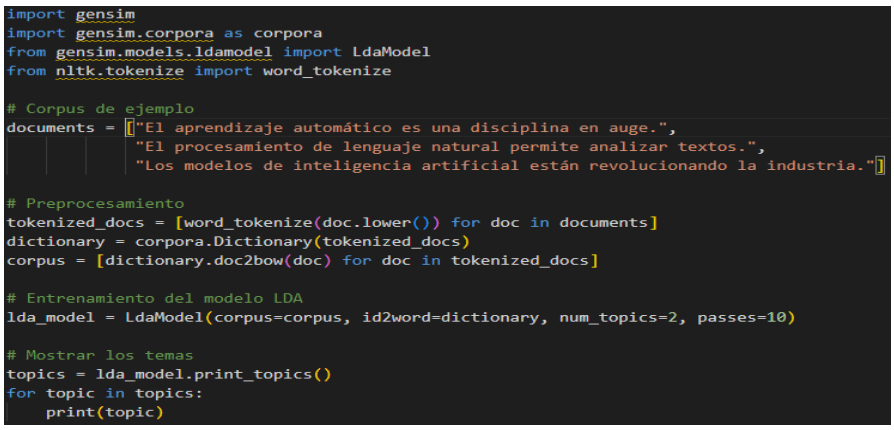
7.5. gensim: Modelado de temas

El modelado de temas es una técnica avanzada de PLN que permite descubrir estructuras latentes en grandes volúmenes de texto. Se basa en la suposición de que los documentos contienen múltiples temas y que cada tema está compuesto por un conjunto de palabras con probabilidades asociadas. Una de las bibliotecas más utilizadas para el modelado de temas en Python es Gensim (Řehůřek, 2025), que proporciona implementaciones eficientes de algoritmos como Latent Dirichlet Allocation, LDA (Kulkarni, 2020) y Non-Negative Matrix Factorization, NMF (Belyadi, 2021). En el siguiente link se puede revisar un acercamiento al tema que topa temas adicionales a los que se revisarán en esta sección: https://youtu.be/IUAHUEy1V0Q?si=nGBpgr0o2nLSl5ai.

Fundamentos del Modelado de Temas con Gensim

Gensim es una biblioteca diseñada para el procesamiento de grandes corpus de texto, permitiendo la extracción de temas sin necesidad de una preetiquetación manual. Su implementación del modelo LDA es una de las más populares debido a su eficacia en la identificación de patrones semánticos en documentos extensos. En la Figura 5, se presenta un ejemplo de modelado de temas con LDA en Gensim.

Figura 5. Modelado de temas con Gensim



```python
import gensim
import gensim.corpora as corpora
from gensim.models.ldamodel import LdaModel
from nltk.tokenize import word_tokenize

# Corpus de ejemplo
documents = ["El aprendizaje automático es una disciplina en auge.",
             "El procesamiento de lenguaje natural permite analizar textos.",
             "Los modelos de inteligencia artificial están revolucionando la industria."]

# Preprocesamiento
tokenized_docs = [word_tokenize(doc.lower()) for doc in documents]
dictionary = corpora.Dictionary(tokenized_docs)
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

# Entrenamiento del modelo LDA
lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=2, passes=10)

# Mostrar los temas
topics = lda_model.print_topics()
for topic in topics:
    print(topic)
```

Nota: Creado por Diego Ordoñez (2025)

Aplicaciones del Modelado de Temas

El modelado de temas con Gensim es ampliamente utilizado en diversos campos. En el ámbito académico, permite organizar grandes colecciones de documentos científicos mediante la detección automática de sus temas principales. En marketing, se usa para analizar opiniones de clientes y extraer información relevante sobre tendencias del mercado. En la gestión de información empresarial, facilita la clasificación automática de documentos y la extracción de conocimiento de grandes bases de datos textuales.

Beneficios y Limitaciones de Gensim

Gensim destaca por su eficiencia en el manejo de corpus extensos sin necesidad de cargar todos los documentos en memoria. Además, su compatibilidad con formatos optimizados como Word2Vec y Doc2Vec (Chen, 2021) permite integraciones avanzadas con modelos de aprendizaje profundo. Sin embargo, el modelado de temas con LDA presenta algunas limitaciones, como la necesidad de una cuidadosa selección del número de temas y la dependencia de una adecuada limpieza y preprocesamiento de los datos.

8. Introducción a Modelos de Deep Learning para PLN:

8.1. Conceptos básicos de aprendizaje profundo

El aprendizaje profundo (Deep Learning) es una subdisciplina del aprendizaje automático que se basa en redes neuronales artificiales con múltiples capas para modelar y extraer representaciones complejas de datos. Inspirado en la estructura del cerebro humano, este enfoque ha revolucionado el Procesamiento de Lenguaje Natural (PLN), el reconocimiento de imágenes y la generación de texto, entre otras áreas. La relación entre estos dos temas es muy importante, y en el siguiente video se tiene un video de Stanford University que profundiza en el tema: https://youtu.be/OQQ-W_63UgQ?si=4yBEt7sRn6MHxyNm.

Redes Neuronales Artificiales

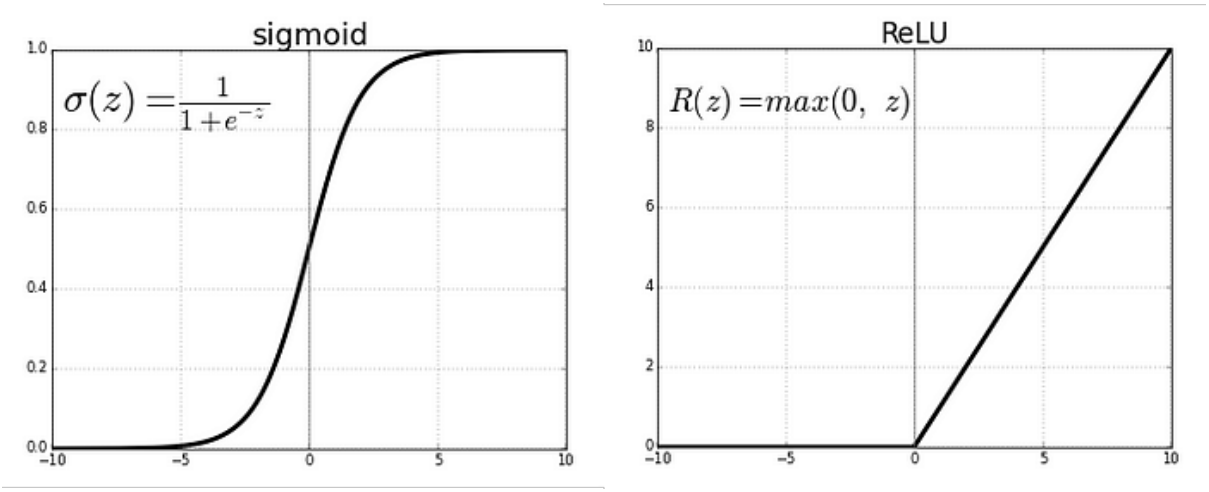
Las redes neuronales artificiales están compuestas por capas de neuronas interconectadas, donde cada neurona procesa una entrada y transmite una salida a las siguientes capas. Las capas principales incluyen:

- Capa de entrada: Recibe los datos iniciales.

- Capas ocultas: Procesan la información a través de combinaciones de pesos y funciones de activación.
- Capa de salida: Genera el resultado final según la tarea deseada.

Las funciones de activación, como ReLU (Rectified Linear Unit) y Sigmoide, introducen no linealidad en el modelo, permitiendo que aprenda patrones complejos (Figura 6).

Figura 6. Sigmoid & ReLU

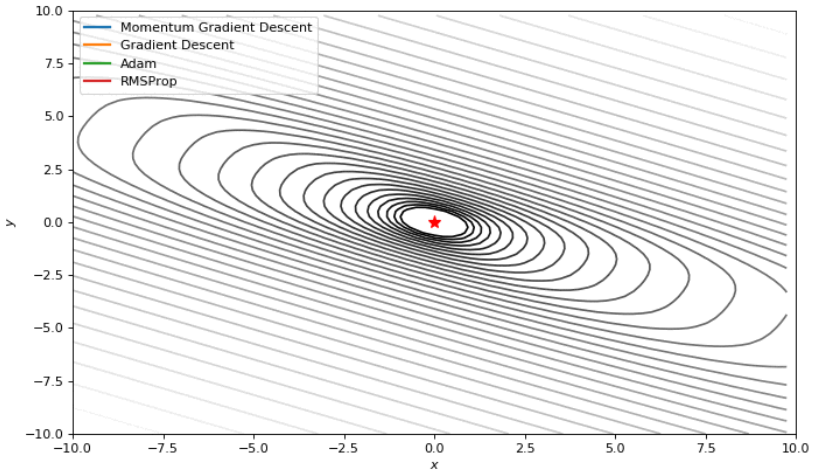


Nota: tomado de Srivastava (2024)

Aprendizaje y optimización

El entrenamiento de una red neuronal implica ajustar los pesos de sus conexiones mediante algoritmos de optimización como Descenso del Gradiente Estocástico (SGD) y Adam (Figura 7). La función de pérdida mide la discrepancia entre las predicciones y los valores reales, y el algoritmo de retropropagación ajusta los pesos para minimizar esta pérdida.

Figura 7. Gradient Descent Methods



Nota: Tomado de Durán (2019)

Aplicaciones en PLN

En el PLN, el aprendizaje profundo ha permitido grandes avances en tareas como la traducción automática, la generación de texto y el análisis de sentimientos. Modelos como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer) han demostrado ser altamente efectivos para comprender el contexto del lenguaje natural y generar respuestas coherentes.

Desafíos y consideraciones

A pesar de sus ventajas, el aprendizaje profundo requiere grandes volúmenes de datos y una alta capacidad computacional. Además, los modelos suelen ser interpretables de manera limitada, lo que plantea desafíos éticos en aplicaciones sensibles como la toma de decisiones automatizada.

El aprendizaje profundo ha transformado el PLN y muchas otras áreas del conocimiento. A medida que se desarrollan nuevas arquitecturas y estrategias de optimización, se espera que continúe evolucionando, permitiendo avances significativos en la comprensión y generación del lenguaje humano.

8.2. Fundamentos de redes neuronales aplicadas a PLN

El uso de redes neuronales en el PLN ha revolucionado la forma en que las computadoras entienden y generan lenguaje humano. Basadas en modelos matemáticos inspirados en el cerebro humano, las redes neuronales pueden captar relaciones complejas dentro de los datos textuales y mejorar tareas como la traducción automática, el análisis de sentimientos y la generación de texto.

Estructura de una red neuronal

Las redes neuronales están compuestas por capas de neuronas artificiales interconectadas. En PLN, se utilizan principalmente redes neuronales profundas (Deep Neural Networks, DNNs), que incluyen:

- **Capa de entrada:** Recibe la representación numérica del texto, como embeddings de palabras.
- **Capas ocultas:** Procesan la información mediante funciones de activación no lineales.
- **Capa de salida:** Genera la predicción deseada, como una clasificación o una respuesta generada.

Las funciones de activación, como ReLU (Rectified Linear Unit) o Softmax, permiten que la red aprenda representaciones complejas del lenguaje y tome decisiones basadas en patrones textuales.

Tipos de redes neuronales en PLN

Redes neuronales recurrentes (RNNs)

Las Redes neuronales recurrentes (RNNs) fueron uno de los primeros enfoques efectivos en PLN. Gracias a su capacidad de procesar secuencias de datos, pueden modelar relaciones contextuales entre palabras. Sin embargo, sufren del problema del gradiente desvanecido, lo que dificulta el aprendizaje en secuencias largas.

LSTM y GRU

Para solucionar las limitaciones de las RNNs estándar, se desarrollaron arquitecturas avanzadas como Long Short-Term Memory (LSTM) y Gated Recurrent Units (GRU), que introducen mecanismos de memoria controlada para retener información relevante en secuencias extensas. Estas redes han demostrado ser eficaces en la traducción automática y el análisis de series temporales de texto.

Redes neuronales convolucionales (CNNs) en PLN

Aunque originalmente diseñadas para el procesamiento de imágenes, las Redes Neuronales Convolucionales (CNNs) también se han utilizado en PLN, particularmente para la clasificación de texto y el análisis de sentimientos. Las CNNs pueden captar patrones locales en frases y oraciones, proporcionando representaciones eficientes del texto.

Transformadores y la revolución en PLN

El desarrollo más significativo en redes neuronales para PLN ha sido el modelo Transformer, introducido en 2017. A diferencia de las RNNs, los Transformadores utilizan mecanismos de atención que procesan todas las palabras de una secuencia simultáneamente, mejorando la eficiencia y la capacidad de captura de dependencias a largo plazo. Modelos basados en esta arquitectura, como BERT y GPT, han establecido nuevos estándares en tareas de PLN, superando los enfoques anteriores en traducción, generación de texto y comprensión del lenguaje.

Las redes neuronales han transformado el campo del PLN, permitiendo avances significativos en la automatización del procesamiento de texto. Con la continua evolución de arquitecturas como los Transformadores y el aprendizaje profundo, el futuro del PLN promete soluciones aún más precisas y sofisticadas en la interacción entre humanos y máquinas.

8.3. Word Embeddings representación densa del lenguaje

En el campo del Procesamiento de Lenguaje Natural (PLN), la representación de palabras es fundamental para que los modelos de aprendizaje profundo comprendan el significado y las relaciones semánticas entre términos. Uno de los enfoques más eficaces para esta tarea es el uso de word embeddings, una técnica que transforma palabras en vectores densos de alta dimensión, preservando relaciones semánticas y sintácticas en un espacio continuo.

De representaciones discretas a representaciones densas

Tradicionalmente, las palabras se representaban mediante esquemas como one-hot encoding, donde cada palabra se asignaba a un vector binario de gran dimensión con una única posición activada. Sin embargo, este enfoque tiene limitaciones, como la alta dimensionalidad y la falta de información sobre relaciones semánticas entre palabras.

Los word embeddings solucionan estos problemas al proyectar palabras en un espacio vectorial denso, donde palabras con significados similares tienen representaciones cercanas. Esta representación permite a los modelos de PLN capturar la semántica del lenguaje de manera más eficiente.

Modelos de word embeddings

Varios modelos han sido desarrollados para generar word embeddings, entre los cuales destacan:

1. Word2Vec: este modelo utiliza redes neuronales para aprender representaciones vectoriales de palabras mediante dos métodos principales:
    - CBOW (Continuous Bag of Words): Predice una palabra a partir de su contexto.
    - Skip-gram: Predice el contexto de una palabra dada.

    En la Figura 8 se presenta un ejemplo de generación de embeddings con Word2Vec.

Figura 8. Generación de embeddings con Word2Vec

```
from gensim.models import Word2Vec
sentences = [
    ['deep', 'learning', 'is', 'powerful'],
    ['word', 'embeddings', 'capture', 'semantics']]
model = Word2Vec(sentences, vector_size=50, window=5, min_count=1, workers=4)
print(model.wv['learning'])
```

Nota: Creado por Diego Ordoñez (2025)

2. **GloVe (Global Vectors for Word Representation):** Este modelo desarrollado por Stanford utiliza una matriz de co-ocurrencias para capturar relaciones semánticas a partir de grandes volúmenes de texto. GloVe es particularmente eficaz para representar analogías y relaciones entre palabras.
3. **FastText:** Una mejora de Word2Vec desarrollada por Facebook, FastText descompone palabras en n-gramas, lo que permite manejar mejor palabras fuera de vocabulario y captar información morfológica relevante.

Aplicaciones de word embeddings en PLN

Los word embeddings han impulsado el desarrollo de modelos más sofisticados en PLN. Algunas de sus aplicaciones incluyen:

- **Análisis de sentimientos:** Permiten mejorar la clasificación de opiniones en redes sociales y reseñas de productos.
- **Traducción automática:** Modelos como seq2seq dependen de embeddings para mejorar la fluidez y coherencia en la traducción de textos.
- **Chatbots y asistentes virtuales:** Facilitan la comprensión del lenguaje natural y la generación de respuestas más precisas.

Los word embeddings han revolucionado la manera en que las máquinas comprenden el lenguaje, proporcionando una base sólida para modelos avanzados de PLN. A medida que la investigación avanza, técnicas más complejas como contextual embeddings (ej. BERT y GPT) están llevando la representación del lenguaje a nuevos niveles de precisión y aplicabilidad.

8.4. Modelos de atención y transformadores

Los modelos de atención y transformadores han revolucionado el campo del PLN al mejorar significativamente la eficiencia y precisión en tareas como la traducción automática, la generación de texto y la comprensión del lenguaje. Su introducción ha permitido superar las limitaciones de modelos secuenciales como las redes neuronales recurrentes (RNNs) y las unidades de memoria a largo plazo (LSTM).

El Mecanismo de atención

El mecanismo de atención se basa en la idea de que, en una secuencia de texto, algunas palabras son más relevantes que otras para la tarea de predicción. En lugar de procesar el texto de manera lineal, la atención asigna pesos a cada palabra de la secuencia, permitiendo que el modelo se enfoque en las partes más importantes.

El mecanismo de atención se puede expresar matemáticamente mediante la formula de la Figura 9, donde:

- QQ (Queries), KK (Keys) y VV (Values) representan proyecciones lineales de los datos de entrada.
- dkd_k es la dimensión de las claves, usada para escalamiento.

Figura 9. Mecanismo de atención

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Nota: Creado por Diego Ordoñez (2025)

Este enfoque permite modelar relaciones a largo plazo en el texto sin depender de estructuras secuenciales rígidas.

Los transformadores y su arquitectura

El modelo Transformer, adopta exclusivamente el mecanismo de atención, eliminando la necesidad de recurrencias. Su arquitectura se compone de:

1. **Capa de embedding:** Convierte palabras en representaciones vectoriales densas.
2. **Múltiples cabezales de atención:** Permiten capturar distintos aspectos del significado de una palabra en paralelo.
3. **Capa feedforward:** Procesa las salidas de la atención con funciones no lineales.
4. **Codificadores y decodificadores:** La arquitectura del Transformer consiste en múltiples capas de codificación y decodificación apiladas, donde los codificadores extraen características del texto de entrada y los decodificadores generan la salida correspondiente.

En la Figura 10 puede verse un pequeño ejemplo de implementación con BERT en Python:

Figura 10. Ejemplo de implementación con BERT

```python
from transformers import AutoModel, AutoTokenizer

modelo = AutoModel.from_pretrained("bert-base-uncased")
tokenizador = AutoTokenizer.from_pretrained("bert-base-uncased")
texto = "Los modelos de atención son clave en PLN."
entrada = tokenizador(texto, return_tensors="pt")
salida = modelo(**entrada)
print(salida)
```

Nota: Creado por Diego Ordoñez (2025)

Aplicaciones de los modelos de atención

Los transformadores han permitido avances notables en diversas aplicaciones del PLN:

- **BERT (Bidirectional Encoder Representations from Transformers):** Utilizado en tareas de clasificación de texto, búsqueda semántica y respuesta a preguntas.

- **GPT (Generative Pre-trained Transformer):** Especializado en generación de texto natural y modelado del lenguaje.
- **T5 (Text-to-Text Transfer Transformer):** Capaz de reformular múltiples tareas de PLN como problemas de generación de texto.

Referencias

Belyadi, H., & Haghighat, A. (2021*). Chapter 3—Machine learning workflows and types.* In H. Belyadi & A. Haghighat (Eds.), Machine Learning Guide for Oil and Gas Using Python (pp. 97–123). Gulf Professional Publishing. https://doi.org/10.1016/B978-0-12-821929-4.00001-9

Chen, Q., & Sokolova, M. (2021*). Specialists, Scientists, and Sentiments: Word2Vec and Doc2Vec in Analysis of Scientific and Medical Texts*. SN Computer Science, 2(5), 414. https://doi.org/10.1007/s42979-021-00807-1

DeepL. (2025). https://www.deepl.com/

Durán, J. (2019). *Todo lo que Necesitas Saber sobre el Descenso del Gradiente Aplicado a Redes Neuronales*. Medium. https://medium.com/metadatos/todo-lo-que-necesitas-saber-sobre-el-descenso-del-gradiente-aplicado-a-redes-neuronales-19bdbb706a78

Explosion. (2025). *spaCy*. https://spacy.io/

Google. (2025). *Google Translate*. https://translate.google.com/

Hugging Face. (2025). https://huggingface.co/docs/transformers/en/index

Kulkarni, S., & Rodd, S. F. (2020). *Context Aware Recommendation Systems: A review of the state of the art techniques*. Computer Science Review, 37, 100255. https://doi.org/10.1016/j.cosrev.2020.100255

Loria, S. (2025). *TextBlob: Simplified Text Processing*. https://textblob.readthedocs.io/

Microsoft. (2025). *MarianNMT*. https://marian-nmt.github.io/

NLTK Project. (2024). *Natural Language Toolkit*. https://www.nltk.org/

Python Software Foundation. (2025). https://www.python.org/

Řehůřek, R. (2025). *Gensim*. https://radimrehurek.com/gensim/

Srivastava, S. (2024). *Understanding the Difference Between ReLU and Sigmoid Activation Functions in Deep Learning.* Medium. https://medium.com/@srivastavashivansh8922/understanding-the-difference-between-relu-and-sigmoid-activation-functions-in-deep-learning-33b280fc2071

Términos para definición

One-hot encoding

One-hot encoding es una técnica de representación de datos utilizada en aprendizaje automático y procesamiento de lenguaje natural (PLN) para convertir variables categóricas en una forma numérica comprensible para los algoritmos. Consiste en transformar cada categoría en un vector binario en el que solo una posición tiene el valor 1 y el resto son 0. Por ejemplo, si tenemos tres categorías: "rojo", "azul" y "verde", la codificación one-hot las representaría como:
- Rojo → [1, 0, 0]
- Azul → [0, 1, 0]
- Verde → [0, 0, 1]

Este método es útil para evitar que los algoritmos interpreten relaciones inexistentes entre categorías cuando se usan representaciones numéricas directas. Sin embargo, presenta limitaciones, como el alto consumo de memoria, ya que para grandes vocabularios o conjuntos de datos genera matrices dispersas con muchos ceros. En PLN, ha sido reemplazado en gran parte por word embeddings, que ofrecen representaciones más densas y ricas en información semántica.

Función de activación

Una función de activación es un componente esencial en una red neuronal artificial que introduce no linealidad en el modelo, permitiendo que la red aprenda relaciones complejas en los datos. Su función principal es transformar la suma ponderada de las entradas de una neurona en una salida que pueda ser utilizada por la siguiente capa de la red. Sin esta transformación, las redes neuronales solo podrían modelar relaciones lineales, limitando su capacidad para resolver problemas complejos como el reconocimiento de imágenes o el procesamiento de lenguaje natural.

Existen diversas funciones de activación, entre las más comunes están la ReLU (Rectified Linear Unit), que activa valores positivos y deja los negativos en cero, la sigmoide, que convierte valores en una probabilidad entre 0 y 1, y la tangente hiperbólica (tanh), que escala los valores entre -1 y 1. La elección de la función de activación adecuada influye en la velocidad de aprendizaje y el rendimiento general del modelo.

PUCE Virtual

EDUCACIÓN FLEXIBLE

La excelencia no se improvisa

síguenos