Práctica Calificada 3 CC0C2

Fecha de entrega: 15 de noviembre

Puntaje máximo: 20 puntos

Entrega del proyecto: 8 puntos

Exposición del proyecto: 12 puntos

Debes presentar un repositorio donde se encuentre todos tus resultados.

Instrucciones generales:

1. Los proyectos serán asignados por orden de elección, asegurando que cada estudiante trabaje en un proyecto diferente.

3. La fecha límite para la entrega del proyecto es el 14 de noviembre. El incumplimiento de la fecha implica una calificación de 0.

4. Las exposiciones tendrán lugar en la misma fecha de entrega. Se asignarán 15 minutos a cada grupo para exponer su proyecto, seguidos de 5 minutos para preguntas.

Proyectos disponibles:

1. Integración de subword embeddings en redes neuronales recurrentes (RNNs) para procesamiento de lenguaje natural

Descripción: Explora cómo los subword embeddings pueden mejorar el manejo de vocabularios abiertos y reducir problemas con palabras fuera del vocabulario (OOV) cuando se integran en arquitecturas RNN.

Componentes clave:

- Subword Embeddings
- Redes Neuronales Recurrentes (RNNs)
- Aplicaciones en NLP (por ejemplo, clasificación de texto)

Componente de código: Implementación en **PyTorch** de un modelo RNN que utiliza subword embeddings para una tarea de clasificación de texto. Incluye la preprocesamiento de datos para generar subword tokens.

2. Cuantización de vectores de palabras en arquitecturas RNN profundas para optimización de modelos

Descripción: Analiza cómo la cuantización de vectores de palabras puede reducir el tamaño del modelo y mejorar la eficiencia computacional en redes RNN profundas sin sacrificar precisión.

Componentes clave:

- Word Vector Quantization
- Deep RNN Architectures
- Optimización de modelos

Componente de código: Código en **PyTorch** que aplica técnicas de cuantización a los vectores de palabras antes de entrenar una arquitectura RNN profunda para una tarea de predicción de secuencia

3. Embeddings de frases en redes RNN bidireccionales para análisis de sentimiento

Descripción: Utiliza sentence embeddings combinados con RNN bidireccionales para capturar contextos más ricos en tareas de análisis de sentimiento.

Componentes clave:

- Sentence Embeddings
- Bidirectional RNNs
- Análisis de sentimiento

Componente de código: Implementación de un modelo bidirectional RNN en **PyTorch** que incorpora sentence embeddings para clasificar el sentimiento de textos en un conjunto de datos de reseñas.

4. Retrofitting de concept embeddings con RNNs para mejorar la comprensión semántica

Descripción: Explora cómo el retrofitting de concept embeddings con lexicons semánticos puede enriquecer las representaciones semánticas en modelos RNN, mejorando tareas como el reconocimiento de entidades.

Componentes clave:

- Concept embeddings
- Retrofitting with semantic lexicons
- RNNs para comprensión semántica

Componente de código: Código en **PyTorch** que realiza retrofitting de concept embeddings y los integra en una RNN para una tarea de reconocimiento de entidades nombradas (NER).

5. Embeddings Gaussianos y RNNs para modelado de incertidumbre en secuencias de texto

Descripción: Combina gaussian embeddings con RNNs para capturar la incertidumbre y variabilidad en la generación de secuencias de texto, mejorando la robustez del modelo.

Componentes clave:

- Gaussian Embeddings
- RNNs
- Modelado de Incertidumbre

Componente de código: Implementación en **PyTorch** de una RNN que utiliza gaussian embeddings para generar texto de manera probabilística, permitiendo la captura de múltiples posibles continuaciones de una secuencia.

6. Embeddings hiperbólicos en redes neuronales recurrentes para representaciones jerárquicas

Descripción: Utiliza hyperbolic embeddings dentro de RNNs para modelar estructuras jerárquicas en datos de secuencia, como árboles sintácticos en procesamiento de lenguaje natural.

Componentes clave:

- Hyperbolic embeddings
- RNNs
- Representaciones jerárquicas

Componente de código: Código en **PyTorch** que implementa una RNN con hyperbolic embeddings para una tarea de clasificación de texto que involucra estructuras jerárquicas.

7. Cuantización de vectores y regularización en RNNs para mitigar el problema del gradiente desaparecido

Descripción: Combina técnicas de word vector quantization con métodos de regularización en RNNs para abordar el problema del gradiente desvanecido, mejorando el entrenamiento de redes profundas.

Componentes clave:

- Word Vector Quantization

- Regularización en RNNs
- Problema del gradiente desaparecido

Componente de código: Implementación en **PyTorch** de una RNN que aplica cuantización de vectores y técnicas de regularización (como dropout o L2) para una tarea de predicción de secuencia.

8. Embeddings de sentencias en arquitecturas de RNN con mecanismos de atención para traducción automática

Descripción: Integra sentence embeddings con mecanismos de atención en modelos RNN para mejorar la precisión y coherencia en tareas de traducción automática.

Componentes clave:

- Sentence embeddings
- Atención en RNNs
- Traducción Automática

Componente de código: Código en **PyTorch** que implementa un modelo de traducción automática con RNN y mecanismo de atención, utilizando sentence embeddings para representar oraciones de entrada y salida.

9. Redes RNN Ppofundas con retrofitting de concept embeddings para análisis semántico

Descripción: Combina arquitecturas RNN profundas con concept embeddings retrofitted para realizar análisis semántico más preciso en tareas complejas como la inferencia de texto.

Componentes clave:

- Deep RNN Architectures
- Concept embeddings
- Retrofitting with semantic lexicons
- Análisis semántico

Componente de código: Implementación en **PyTorch** de una RNN profunda que utiliza concept embeddings retrofitted para una tarea de inferencia de texto (NLI).

10. Comparación de embeddings hiperbólicos y Gaussianos en modelos Transformer y RNNs para tareas secuenciales

Descripción: Analiza y compara el rendimiento de hyperbolic embeddings y gaussian embeddings en modelos basados en Transformers y RNNs, evaluando su efectividad en diversas tareas secuenciales.

Componentes clave:

- Hyperbolic embeddings
- Gaussian embeddings
- Transformer networks vs. RNNs
- Tareas secuenciales

Componente de código: Código en **PyTorch** que implementa tanto un modelo Transformer como una RNN, cada uno utilizando hyperbolic y gaussian embeddings, para una tarea de secuencia como la generación de texto o la traducción. Incluye métricas de comparación de rendimiento.

Rúbricas de evaluación:

1. Entrega del proyecto (8 puntos):

La entrega debe incluir el código fuente, la documentación del proyecto, y los resultados de las pruebas realizadas con el corpus asignado.

Criterio	Puntos	Descripción
Funcionalidad del código	3	El código debe implementar correctamente el proyecto propuesto y ser completamente funcional, sin errores que afecten su desempeño.
Eficiencia del algoritmo	2	El código debe demostrar eficiencia en el procesamiento, especialmente en proyectos que tratan con grandes volúmenes de datos.
Claridad y estructura del código	1.5	El código debe estar bien estructurado, con comentarios claros y buenas prácticas de programación (modularización, nombres descriptivos, etc.).
Documentación del proyecto	1.5	La documentación debe explicar la implementación, las decisiones técnicas y cómo ejecutar el proyecto correctamente.

2. Exposición del proyecto (12 puntos):

Cada grupo tendrá 15 minutos para exponer su proyecto, seguidos de 5 minutos de preguntas y respuestas.

Criterio	Puntos	Descripción
Claridad en la explicación	4	El grupo debe explicar el proyecto de manera clara, estructurada y coherente, destacando los aspectos clave de su implementación.
Entendimiento técnico	3	El grupo debe demostrar un entendimiento profundo de los conceptos aplicados en el proyecto (ej: tokenización, modelos n-grama, suavizado).
Resultados y análisis	3	El grupo debe presentar los resultados obtenidos de manera clara, con análisis crítico sobre el rendimiento del modelo o algoritmo implementado.
Manejo de preguntas	2	El grupo debe ser capaz de responder a las preguntas de los compañeros o del profesor de manera adecuada y demostrando comprensión del tema.