Procesamiento de Lenguaje Natural
Tarea 4

Rayo Mosquera, Jhon Stewar
j.rayom@uniandes.edu.co

De La Rosa Peredo, Carlos Raul
c.delarosap@uniandes.edu.co

Mario Garrido Córdoba
m.garrido10@uniandes.edu.co

Octubre 2024

Punto I

Procesamos una colección de libros de Jane Austen, Leo Tolstoy y James Joyce obtenidos del Proyecto Gutenberg, para entrenar embeddings de palabras usando Word2Vec de Gensim. Comenzamos leyendo el texto crudo de cada libro y aplicamos pasos de preprocesamiento como la eliminación de puntuación, la tokenización, la conversión a minúsculas y la eliminación de palabras vacías. Estos pasos aseguran que el corpus esté bien preparado para el entrenamiento de embeddings. Una vez preprocesado, guardamos el texto tokenizado para su uso futuro.

Después de procesar y tokenizar los libros, combinamos todos los textos en un solo corpus y entrenamos modelos Word2Vec con tres tamaños de vector: 50, 100 y 300 dimensiones. Configuramos el entrenamiento con parámetros como el tamaño de la ventana de contexto, la cantidad mínima de apariciones de una palabra y el número de épocas. Cada modelo entrenado se guarda en disco utilizando una convención de nombres específica que incluye nuestro código de grupo.

Punto II

Cargamos modelos preentrenados de Word2Vec para los libros, enfocándonos en los personajes principales y recuperando las palabras más semánticamente similares a ellos. Después de extraer los vectores de palabras relevantes, reducimos su dimensionalidad y los graficamos en dos dimensiones para visualizarlos. También exploramos las relaciones temáticas de conceptos clave como “amor”, “guerra”, “paz”, “muerte” y “honor” mediante razonamiento analógico, obteniendo una comprensión más profunda de los temas en estos libros.

Estrategia para Graficar Embeddings en Dos Dimensiones

Usamos t-SNE (t-distributed Stochastic Neighbor Embedding) como método principal para reducir la dimensionalidad de los vectores de palabras generados por los modelos Word2Vec. Este método captura estructuras locales en los datos, asegurando que las palabras semánticamente cercanas se mantengan próximas en el gráfico bidimensional.

También hemos implementado PCA (Análisis de Componentes Principales) como alternativa, aunque t-SNE es nuestro método preferido debido a su efectividad en mantener relaciones sutiles entre palabras. Después de la reducción de dimensionalidad, creamos gráficos de dispersión donde cada punto representa una palabra, y su proximidad refleja similitud semántica. Los gráficos están anotados con etiquetas de palabras para interpretar visualmente las relaciones entre personajes y términos relacionados.

Esta estrategia nos permite visualizar cómo los personajes y sus palabras más similares se organizan en un espacio semántico de baja dimensionalidad, ayudándonos a identificar agrupaciones temáticas y analizar relaciones entre diferentes libros.

Discusión de las Relaciones Encontradas Usando Razonamiento Analógico

Exploramos temas como "amor", "guerra", "paz", "muerte" y "honor" a través del razonamiento analógico, descubriendo lo siguiente:

- **Amor**: El "amor" está asociado con palabras como "casado", "celoso", "felicidad" y "encanto". En Tolstoy, vemos asociaciones específicas con personajes como "Sonetchka", mientras que Joyce se enfoca en aspectos más internos del amor, con términos como "amante".
- **Guerra**: En Tolstoy, la "guerra" está vinculada a contextos históricos y nacionales, con palabras como "patriótico" y "heroísmo". En Austen, la "guerra" se usa más metafóricamente para representar conflictos sociales o personales, como "sociedad" y "asunto".
- **Paz**: La "paz" en Tolstoy está asociada con luchas filosóficas, mientras que en Austen tiene un trasfondo emocional, vinculada a palabras como "miserable" y "confianza".
- **Muerte**: La "muerte" evoca asociaciones emocionales y espirituales fuertes, con palabras como "dolor", "ángel" y "enfermedad". En Tolstoy, está más relacionada con reflexiones morales, mientras que en Austen se centra más en la pérdida personal.
- **Honor**: En Austen, el "honor" está relacionado con la reputación social y el comportamiento personal, mientras que en Tolstoy está ligado al deber y la rivalidad. En Joyce, se asocia con conflictos internos y la identidad.

A través de estas analogías, logramos comprender mejor cómo cada autor trata estos temas universales de manera única, reflejando la riqueza y profundidad de sus obras literarias.

Punto III

Entrenamos y evaluamos redes neuronales feed-forward (FFNN) para la atribución de autor utilizando embeddings personalizados de Word2Vec. Procesamos los datos, construimos tres modelos con diferentes complejidades y evaluamos su rendimiento con tres tamaños de embeddings (50, 100 y 300 dimensiones).

Comenzamos cargando y preprocesando los textos. Se extraen oraciones de 150 a 250 palabras y se almacenan en un DataFrame con las columnas `autor` y `oración`. Luego, dividimos los datos en conjuntos de entrenamiento (70%), validación (15%) y prueba (15%). Utilizamos un tokenizador para convertir las oraciones en secuencias de enteros, y las secuencias se rellenan para garantizar una longitud uniforme.

A continuación, cargamos los modelos Word2Vec personalizados con 50, 100 y 300 dimensiones. Para cada tamaño de embedding, creamos una capa de embedding en Keras que mapea las palabras tokenizadas a vectores, y esta capa sirve como la entrada para los modelos.

Definimos tres arquitecturas FFNN:

- **Modelo 1** (El más simple): Capa de embedding → Capa de flatten → Capa densa (128 unidades) → Capa de salida (3 unidades).
- **Modelo 2** (Intermedio): Capa de embedding → Capa de flatten → Capas densas (256, 128 unidades) → Capa de salida.
- **Modelo 3** (El más complejo): Capa de embedding → Capa de flatten → Capas densas (512, 256, 128 unidades) → Capa de salida.

Entrenamos y evaluamos cada modelo durante 10 épocas con un tamaño de lote de 32. Evaluamos cada combinación de modelo y tamaño de embedding en el conjunto de prueba, midiendo precisión, exactitud y recall. El notebook muestra la arquitectura del modelo y los resultados de rendimiento final para cada configuración.

Al explorar diferentes arquitecturas FFNN y tamaños de embeddings para la atribución de autor, encontramos que los modelos más simples y los embeddings más pequeños tienden a generalizar mejor, siendo el Modelo 1 con embeddings de 50 dimensiones el que ofrece el mejor rendimiento general.

Modelo	**Tamaño del Embedding**	**Exactitud**	**Precisión**	**Recall**
Modelo 1	50	91.32%	90.49%	87.50%
Modelo 2	50	91.32%	90.23%	86.58%
Modelo 3	50	90.50%	88.69%	86.59%
Modelo 1	100	89.26%	88.02%	83.98%
Modelo 2	100	90.08%	88.72%	87.45%
Modelo 3	100	89.67%	88.31%	86.53%
Modelo 1	300	90.91%	90.18%	86.14%
Modelo 2	300	90.50%	89.27%	86.25%
Modelo 3	300	89.67%	88.70%	85.34%

Tabla 1: Comparación de resultados de los modelos FFNN con diferentes tamaños de embeddings.

Resumen de Resultados

A partir de la evaluación, observamos que el Modelo 1 (la arquitectura más simple) con embeddings de 50 dimensiones logra el mejor rendimiento general, con una exactitud del 91.32%, precisión del 90.49% y *recall del 87.50%*. Esto sugiere que una arquitectura más simple combinada con embeddings de menor dimensionalidad generaliza mejor a los datos.

A medida que aumenta la complejidad del modelo (por ejemplo, el Modelo 3 con 512 unidades), el rendimiento tiende a disminuir ligeramente. El Modelo 3 con embeddings de 50 dimensiones obtiene una exactitud del 90.50%, precisión del 88.69% y *recall del 86.59%*. Esto indica que las arquitecturas más complejas pueden ser más propensas al sobreajuste, especialmente cuando el conjunto de datos no es lo suficientemente grande para justificar los parámetros adicionales.

Al aumentar las dimensiones de los embeddings (100 y 300), notamos una ligera disminución en el rendimiento. Por ejemplo, el Modelo 1 con embeddings de 100 dimensiones alcanza una exactitud del 89.26%, mientras que con 300 dimensiones alcanza 90.91%. Sin embargo, los modelos más complejos con 300 dimensiones no logran una mejora significativa y tienen un mayor riesgo de sobreajuste.

Análisis

Los resultados muestran claramente que los modelos más simples con embeddings de menor dimensionalidad son más adecuados para la tarea de atribución de autor con este conjunto de datos. El Modelo 1 supera consistentemente a las arquitecturas más complejas, y los embeddings de 50 dimensiones proporcionan el rendimiento más equilibrado en términos de exactitud, precisión y *recall*. Aumentar la complejidad del modelo o el tamaño del embedding genera rendimientos decrecientes, probablemente debido al sobreajuste o la dificultad añadida de generalizar con embeddings de mayor dimensionalidad.

Para mejorar el rendimiento, podríamos explorar técnicas como dropout o regularización L2 para reducir el sobreajuste en los modelos más complejos. Además, la validación cruzada proporcionaría una estimación más confiable del rendimiento del modelo en diferentes divisiones de los datos. Sin embargo, basándonos en estos resultados, el Modelo 1 con embeddings de 50 dimensiones sigue siendo la combinación más efectiva para esta tarea de atribución de autor.

Punto IV

Entrenamos y evaluamos redes neuronales feed-forward (FFNN) para la atribución de autor utilizando embeddings pre-entrenados de GloVe. Procesamos los datos, construimos tres modelos con diferentes complejidades y evaluamos su rendimiento utilizando embeddings de GloVe con tres tamaños: 50, 100 y 300 dimensiones.

Comenzamos cargando el dataset de oraciones etiquetadas, similar al utilizado en el Punto III. El dataset se divide en entrenamiento (70%), validación (15%) y prueba (15%). Luego, las oraciones se tokenizan y se rellenan para garantizar una longitud uniforme. Para cada tamaño de embedding, creamos una capa de embeddings en Keras que utiliza los vectores pre-entrenados de GloVe como entrada.

Definimos tres arquitecturas FFNN:

- **Modelo 1** (El más simple): Capa de embedding → Capa de flatten → Capa densa (128 unidades) → Capa de salida (3 unidades).
- **Modelo 2** (Intermedio): Capa de embedding → Capa de flatten → Capas densas (256, 128 unidades) → Capa de salida.
- **Modelo 3** (El más complejo): Capa de embedding → Capa de flatten → Capas densas (512, 256, 128 unidades) → Capa de salida.

Cada modelo se entrena durante 10 épocas con un tamaño de lote de 32. Evaluamos cada combinación de modelo y tamaño de embedding en el conjunto de prueba, midiendo precisión, exactitud y *recall*. A continuación, presentamos los resultados obtenidos.

Modelo	**Tamaño del Embedding**	**Exactitud**	**Precisión**	**Recall**
Modelo 1	50	73.55%	77.88%	62.24%
Modelo 2	50	71.90%	77.03%	60.49%
Modelo 3	50	70.66%	75.24%	59.72%
Modelo 1	100	71.07%	74.25%	60.53%
Modelo 2	100	69.83%	80.01%	57.06%
Modelo 3	100	69.83%	77.90%	56.83%
Modelo 1	300	68.60%	87.40%	52.81%
Modelo 2	300	70.25%	75.50%	56.18%
Modelo 3	300	73.14%	85.68%	60.74%

Tabla 2: Comparación de resultados de los modelos FFNN con embeddings pre-entrenados de GloVe.

Resumen de Resultados

Observamos que los modelos más simples tienden a obtener mejores resultados que las arquitecturas más complejas. El Modelo 1 con embeddings de 50 dimensiones obtiene la mayor exactitud general (73.55%) y una precisión de 77.88%, pero su *recall* es más bajo (62.24%), lo que sugiere que el modelo comete más falsos negativos.

Los modelos más complejos (por ejemplo, el Modelo 3) no ofrecen un rendimiento superior y, en algunos casos, sufren de sobreajuste. Por ejemplo, con embeddings de 300 dimensiones, el Modelo 3 alcanza una exactitud del 73.14%, pero su *recall* cae a 60.74%.

En general, notamos que los embeddings más pequeños (50 dimensiones) producen mejores resultados, mientras que los embeddings más grandes introducen ruido adicional, lo que dificulta la generalización del modelo.

Análisis

Los resultados sugieren que los modelos más simples y embeddings de menor dimensionalidad son más efectivos para esta tarea, probablemente porque los datos no son lo suficientemente grandes como para justificar modelos más complejos o embeddings de alta dimensión.

Para mejorar el rendimiento, recomendamos explorar técnicas de regularización como *dropout* y validación cruzada para reducir el sobreajuste. En particular, el uso de **early stopping** podría evitar que los modelos entrenen innecesariamente durante 10 épocas. A pesar de estos desafíos, el Modelo 1 con 50 dimensiones ofrece el mejor equilibrio entre precisión y simplicidad, por lo que es la opción más efectiva para esta tarea de atribución de autor.

Punto V

Comparamos los resultados obtenidos en las secciones anteriores, analizando los efectos del uso de embeddings preentrenados frente a embeddings personalizados en el corpus, así como el impacto de la dimensionalidad de los embeddings.

¿Es mejor utilizar embeddings preentrenados o personalizados en el corpus?

Los resultados muestran que tanto Los embeddings personalizados (entrenados con los textos de los tres autores) como los preentrenados (GloVe) tienen ventajas y desventajas:

- **Embeddings personalizados:** Capturan con mayor precisión las características estilísticas del corpus, ofreciendo un rendimiento superior, especialmente en los textos de Jane Austen, donde los modelos alcanzan un alto *recall* y precisión. Esto se debe a que los embeddings personalizados se entrenan directamente con el vocabulario y estilo del corpus, adaptándose mejor al contexto literario específico.
- **Embeddings preentrenados:** Aunque ofrecen un conocimiento semántico más amplio, no se adaptan bien a estilos literarios complejos como los de James Joyce, donde los modelos muestran un menor *recall*. Sin embargo, los embeddings preentrenados de GloVe de 300 dimensiones producen resultados competitivos, especialmente en modelos más complejos, lo que sugiere que pueden ser útiles si se aplican técnicas adicionales como *fine-tuning*.

En conclusión, para tareas de atribución de autor con corpus literarios, los embeddings personalizados ofrecen mejores resultados, ya que capturan las peculiaridades estilísticas del texto. Los embeddings preentrenados pueden ser útiles cuando el corpus tiene un vocabulario más diverso o se dispone de técnicas adicionales de *fine-tuning*.

Impacto de la dimensionalidad de los embeddings

- **Embeddings de 50D:** Ofrecen un rendimiento equilibrado, evitando el sobreajuste y proporcionando una precisión competitiva, especialmente en modelos más simples. Estos embeddings son más eficientes en tareas donde no es necesario capturar relaciones semánticas excesivamente complejas.
- **Embeddings de 100D:** Muestran un rendimiento intermedio, aunque sin mejoras significativas sobre los embeddings de 50D. Los modelos con estos embeddings capturan más información relevante, pero la diferencia en precisión es marginal, lo que indica que la mayor dimensionalidad no siempre se traduce en una mejora significativa.

- **Embeddings de 300D:** Producen el mejor rendimiento en términos de precisión, como se observa en el Modelo 3 del Punto IV, que logra una precisión del 85.68%. Sin embargo, la mayor dimensionalidad también conduce a sobreajuste, particularmente en arquitecturas complejas, afectando la generalización.

En conclusión, aunque los embeddings de 300D son útiles para capturar más información semántica, los embeddings de menor dimensión (50D) son más eficientes y ofrecen un mejor equilibrio entre simplicidad y rendimiento, especialmente para evitar el sobreajuste.

Conclusiones y Recomendaciones

Con base en los resultados obtenidos:

- Los **embeddings personalizados son más adecuados** para tareas de atribución de autor en corpus literarios específicos, ya que capturan mejor las características estilísticas.
- Los **embeddings preentrenados pueden ser útiles** si se aplican técnicas de *fine-tuning* o si se trabaja con textos más variados.
- Los **embeddings de menor dimensión (50D)** ofrecen el mejor equilibrio entre simplicidad y rendimiento, evitando el sobreajuste.
- Los **embeddings de 300D** muestran un rendimiento superior en términos de precisión, pero requieren mayor cuidado para evitar el sobreajuste mediante regularización y técnicas como *early stopping*.

En resumen, los embeddings personalizados, combinados con modelos más simples y embeddings de 50 dimensiones, ofrecen la solución más efectiva para esta tarea específica. Sin embargo, explorar arquitecturas más avanzadas, como transformers, o utilizar embeddings ajustados al dominio literario podría mejorar aún más los resultados.