Manifold Flow Reduction (MFR): Una Metodología Rigurosa para la Reducción de Dimensionalidad No Lineal en Modelos Basados en Ecuaciones Diferenciales Ordinarias

Kitsu
Institución Académica
correo@ejemplo.com

November 4, 2024

Abstract

En este trabajo se introduce **Manifold Flow Reduction (MFR)**, una metodología rigurosa y novedosa para la reducción de dimensionalidad no lineal de embeddings en modelos de Procesamiento de Lenguaje Natural (NLP) basados en Ecuaciones Diferenciales Ordinarias (ODEs), como **LiquidNLP**. MFR combina conceptos avanzados de teoría de manifolds, flujos normales (*Normalizing Flows*) y dinámicas continuas para transformar embeddings de alta dimensionalidad en representaciones latentes de baja dimensionalidad, preservando la estructura intrínseca y las relaciones no lineales de los datos. Se presenta una formulación matemática exhaustiva de MFR, junto con un análisis teórico de sus propiedades fundamentales y su integración detallada en el contexto de LiquidNLP. Además, se introduce una serie de mejoras en la arquitectura del modelo, incluyendo neuronas adaptativas y mecanismos de atención personalizados, que potencian aún más la eficiencia y capacidad de generalización del sistema. Los resultados experimentales demuestran que MFR supera a métodos convencionales en términos de eficiencia computacional y capacidad de captura de relaciones semánticas complejas.

Contents

1 Introducción

La reducción de dimensionalidad es una tarea fundamental en el campo del Aprendizaje Automático y, específicamente, en el Procesamiento de Lenguaje Natural (NLP). En modelos de NLP modernos, los embeddings de palabras o tokens son representaciones vectoriales de alta dimensionalidad que capturan información semántica y contextual. Sin embargo, estas representaciones pueden ser costosas en términos de memoria y computación, especialmente cuando se manejan secuencias largas o vocabularios extensos.

Métodos tradicionales como la Transformada Rápida de Fourier (FFT), la Descomposición en Valores Singulares (SVD) y Autoencoders Variacionales (VAEs) han sido utilizados para la

reducción de dimensionalidad. No obstante, estos enfoques presentan limitaciones en la captura de relaciones no lineales complejas presentes en los embeddings de texto.

En este contexto, presentamos **Manifold Flow Reduction (MFR)**, una metodología innovadora que combina la teoría de manifolds, flujos normales y dinámicas continuas para lograr una reducción de dimensionalidad eficaz y capaz de preservar las relaciones no lineales intrínsecas en las secuencias de datos. MFR está diseñada para integrarse de manera fluida con modelos basados en Ecuaciones Diferenciales Ordinarias (ODEs), como **LiquidNLP**, mejorando su eficiencia y capacidad de generalización. Además, se incorporan mejoras arquitectónicas que optimizan aún más el rendimiento del modelo.

2 Fundamentos Matemáticos

2.1 Teoría de Manifolds

La teoría de manifolds establece que muchos conjuntos de datos de alta dimensionalidad residen en una **variedad** (*manifold*) de baja dimensionalidad dentro del espacio euclidiano. Formalmente, una variedad $\mathcal{M}$ de dimensión d está embebida en un espacio euclidiano $\mathbb{R}^D$ ($D > d$) si existe una función suave $\phi : \mathcal{M} \to \mathbb{R}^D$ que es un embebimiento.

[Embebimiento] Un *embebimiento* $\phi : \mathcal{M} \to \mathbb{R}^D$ es una función difeomórfica (inversible, diferenciable y cuya inversa también es diferenciable) que mapea la variedad $\mathcal{M}$ en $\mathbb{R}^D$ de manera tal que la estructura diferencial de $\mathcal{M}$ se preserva en la imagen $\phi(\mathcal{M})$.

La reducción de dimensionalidad busca identificar y parametrizar esta variedad para representar los datos de manera más eficiente sin perder información esencial. Técnicas basadas en manifolds aprovechan esta estructura para realizar transformaciones no lineales que capturan la topología intrínseca de los datos.

2.2 Flujos Normales (*Normalizing Flows*)

Los **Flujos Normales** son una clase de modelos generativos que transforman una distribución simple $p_Z(\mathbf{z})$ (por ejemplo, una normal estándar) en una distribución compleja $p_X(\mathbf{x})$ mediante una serie de transformaciones invertibles y diferenciables. Cada transformación f_k en la secuencia de flujos se define de tal manera que la transformación completa $f = f_K \circ \cdots \circ f_1$ es invertible:

$$\mathbf{x} = f(\mathbf{z}) = f_K \circ \cdots \circ f_1(\mathbf{z}) \tag{1}$$

La densidad de $\mathbf{x}$ se puede calcular mediante el cambio de variable:

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \left| \det \left(\frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| \tag{2}$$

2.2.1 Capa de Acoplamiento Afin Compartida

Una de las implementaciones más avanzadas de los flujos normales son las **Capa de Acoplamiento Afin Compartida** (*Shared Affine Coupling Layers*). Estas capas dividen la entrada en dos partes y aplican una transformación affine a una de ellas condicionada por la otra, con la ventaja adicional de compartir parámetros entre múltiples capas de flujo. Esto reduce significativamente el número de parámetros y mejora la eficiencia del modelo.

$$\mathbf{y}_1 = \mathbf{x}_1 \tag{3}$$

$$\mathbf{y}_2 = \gamma \cdot (\mathbf{x}_2 \odot \exp(\alpha \cdot \mathbf{s}(\mathbf{x}_1)) + \beta \cdot \mathbf{t}(\mathbf{x}_1)) + (1 - \gamma) \cdot \mathbf{x}_2 \tag{4}$$

Donde:

- $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$ y $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2]$.
- $\mathbf{s}$ y $\mathbf{t}$ son redes neuronales que aprenden las transformaciones de escala y desplazamiento, respectivamente.

- α y β son adaptadores de escala y desplazamiento.
- γ es una puerta de control que regula la mezcla entre la transformación affine y la identidad.

La inversa de esta transformación es:

$$\mathbf{x}_1 = \mathbf{y}_1 \tag{5}$$

$$\mathbf{x}_2 = (\mathbf{y}_2 \cdot \gamma^{-1} - \beta \cdot \mathbf{t}(\mathbf{y}_1)) \odot \exp(-\alpha \cdot \mathbf{s}(\mathbf{y}_1)) + \mathbf{x}_2 \cdot (1 - \gamma^{-1}) \tag{6}$$

El determinante del Jacobiano J de esta transformación es:

$$\det J = \gamma \cdot \exp\left(\sum_i \alpha \cdot s_i(\mathbf{x}_1)\right) \tag{7}$$

Esta propiedad facilita el cálculo eficiente del log-determinante necesario para la función de pérdida.

2.3 Dinámicas Continuas (*ODEs*)

Las **Ecuaciones Diferenciales Ordinarias** (*ODEs*) describen la evolución de un estado oculto $\mathbf{h}(t)$ a lo largo del tiempo:

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{f}(\mathbf{h}(t), t) \tag{8}$$

donde $\mathbf{f}$ es una función neural que modela las interacciones temporales en el estado oculto. Esta formulación permite capturar dependencias temporales de manera continua y adaptativa, superando las limitaciones de modelos discretos como las Redes Neuronales Recurrentes (*RNNs*).

3 Manifold Flow Reduction (MFR)

3.1 Descripción General

Manifold Flow Reduction (MFR) es una metodología de reducción de dimensionalidad no lineal que transforma embeddings de alta dimensionalidad en representaciones latentes de baja dimensionalidad, preservando la estructura intrínseca y las relaciones no lineales de los datos. MFR se integra de manera fluida con modelos basados en ODEs, como **LiquidNLP**, permitiendo una captura eficiente de dependencias temporales en secuencias de datos. Además, se han introducido mejoras arquitectónicas que optimizan la capacidad de modelado y la eficiencia computacional del sistema.

3.2 Componentes de MFR

MFR se compone de los siguientes elementos fundamentales:

1. **Mapeo No Lineal mediante Flujos Normales Compartidos**: Utiliza una secuencia de **Capa de Acoplamiento Afin Compartida** para transformar embeddings originales $\mathbf{E} \in \mathbb{R}^{B \times S \times D}$ a un espacio latente $\mathbf{Z} \in \mathbb{R}^{B \times S \times d}$, donde $d < D$. La compartición de parámetros entre capas de flujo optimiza el número de parámetros y mejora la eficiencia del modelo.
2. **Dinámica Continua en el Espacio Latente**: Modela la evolución temporal de los embeddings latentes $\mathbf{Z}(t)$ mediante una ODE, capturando dependencias temporales de manera continua.
3. **Reconstrucción y Regularización**: Incorpora mecanismos de reconstrucción y regularización para asegurar que la transformación preserve la información esencial de los embeddings originales.

4. **Neurona Líquida Adaptativa**: Introduce neuronas adaptativas que ajustan dinámicamente sus parámetros en función del contexto actual, mejorando la capacidad de modelado y la eficiencia.

5. **Célula de Tiempo Líquido Mejorada**: Implementa una célula de tiempo líquida que garantiza invariancia a deformaciones temporales, permitiendo un procesamiento más robusto y flexible de las secuencias de datos.

6. **Mecanismo de Atención Personalizado**: Incorpora un mecanismo de atención personalizado que optimiza la captura de relaciones semánticas complejas en las secuencias de datos.

3.3 Formulación Matemática de MFR

3.3.1 Mapeo No Lineal mediante Flujos Normales Compartidos

Sea $\mathbf{E} \in \mathbb{R}^{B \times S \times D}$ la matriz de embeddings, donde B es el tamaño del batch, S es la longitud de la secuencia y D es la dimensión original de los embeddings. MFR define una transformación invertible f_θ que mapea cada embedding $\mathbf{e}_i \in \mathbb{R}^D$ a una representación latente $\mathbf{z}_i \in \mathbb{R}^d$:

$$\mathbf{z}_i = f_\theta(\mathbf{e}_i) \tag{9}$$

Esta transformación se realiza mediante una secuencia de K **Capa de Acoplamiento Afin Compartida**:

$$f_\theta = f_K \circ f_{K-1} \circ \cdots \circ f_1 \tag{10}$$

Donde cada capa f_k es una **Capa de Acoplamiento Afin Compartida** que transforma una parte de los embeddings condicionada por la otra. El determinante del Jacobiano de la transformación completa es la suma de los determinantes de cada capa:

$$\log\det J = \sum_{k=1}^{K} \log\det J_k \tag{11}$$

3.3.2 Dinámica Continua en el Espacio Latente

Una vez obtenida la representación latente $\mathbf{Z}$, la dinámica continua se define mediante una ODE que modela la evolución del estado oculto $\mathbf{h}(t)$:

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{g}_\phi(\mathbf{h}(t), \mathbf{z}(t), t) \tag{12}$$

Donde $\mathbf{g}_\phi$ es una función neural que determina cómo el estado oculto evoluciona en función del estado actual $\mathbf{h}(t)$, la representación latente $\mathbf{z}(t)$ y el tiempo t.

3.3.3 Reconstrucción y Regularización

Para asegurar que la transformación f_θ preserve la información esencial de los embeddings originales, se introduce una función de pérdida que combina una **pérdida de reconstrucción** y una **divergencia de Kullback-Leibler (KL)**:

$$\mathcal{L}_{\text{MFR}} = \mathcal{L}_{\text{reconstrucción}} + \lambda \mathcal{L}_{\text{KL}} \tag{13}$$

Donde:

- $\mathcal{L}_{\text{reconstrucción}} = \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \|\mathbf{e}_{bs} - \hat{\mathbf{e}}_{bs}\|^2$ mide la discrepancia entre los embeddings originales $\mathbf{E}$ y los embeddings reconstruidos $\hat{\mathbf{E}}$.
- $\mathcal{L}_{\text{KL}} = \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \text{KL}(q_\theta(\mathbf{z}_{bs}|\mathbf{e}_{bs})||p(\mathbf{z}_{bs}))$ regulariza la distribución latente para que se asemeje a una distribución prior $p(\mathbf{z})$, generalmente una normal estándar.
- λ es un hiperparámetro que balancea ambas pérdidas.

3.4 Definición de las Clases del Modelo

3.4.1 SharedAffineCouplingLayer

La clase **SharedAffineCouplingLayer** implementa una capa de acoplamiento affine compartida, que es fundamental para la transformación invertible de los embeddings originales a representaciones latentes.

$$\mathbf{y}_1 = \mathbf{x}_1 \tag{14}$$
$$\mathbf{y}_2 = \gamma \cdot (\mathbf{x}_2 \odot \exp(\alpha \cdot \mathbf{s}(\mathbf{x}_1)) + \beta \cdot \mathbf{t}(\mathbf{x}_1)) + (1 - \gamma) \cdot \mathbf{x}_2 \tag{15}$$

Donde:

- $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$ y $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2]$.
- $\mathbf{s}$ y $\mathbf{t}$ son redes neuronales que aprenden las transformaciones de escala y desplazamiento, respectivamente.
- α y β son adaptadores de escala y desplazamiento.
- γ es una puerta de control que regula la mezcla entre la transformación affine y la identidad.

La inversa de esta transformación es:

$$\mathbf{x}_1 = \mathbf{y}_1 \tag{16}$$
$$\mathbf{x}_2 = (\mathbf{y}_2 \cdot \gamma^{-1} - \beta \cdot \mathbf{t}(\mathbf{y}_1)) \odot \exp(-\alpha \cdot \mathbf{s}(\mathbf{y}_1)) + \mathbf{x}_2 \cdot (1 - \gamma^{-1}) \tag{17}$$

El determinante del Jacobiano J de esta transformación es:

$$\det J = \gamma \cdot \exp\left(\sum_i \alpha \cdot s_i(\mathbf{x}_1)\right) \tag{18}$$

3.4.2 OptimizedFlowDimensionalityReduction

La clase **OptimizedFlowDimensionalityReduction** implementa una secuencia de capas de acoplamiento affine compartidas optimizadas para la reducción de dimensionalidad. Incluye mecanismos de proyección y adaptadores que mejoran la eficiencia y la capacidad de modelado del flujo.

$$\mathbf{z} = f_\theta(\mathbf{E}) \in \mathbb{R}^{B \times S \times d} \tag{19}$$

Donde f_θ es una composición de múltiples capas de acoplamiento affine compartidas:

$$f_\theta = f_K \circ f_{K-1} \circ \cdots \circ f_1 \tag{20}$$

Cada capa introduce transformaciones no lineales condicionadas, y la compartición de parámetros entre capas reduce el número total de parámetros del modelo.

3.4.3 ODEFunc

La clase **ODEFunc** define la función que describe la dinámica del estado oculto en el espacio latente. Utiliza normalización por capas para mejorar la estabilidad y capacidad de generalización del modelo.

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{g}_\phi(\mathbf{h}(t), t) \tag{21}$$

Donde $\mathbf{g}_\phi$ está parametrizada por una red neuronal que incluye normalización por capas:

$$\mathbf{g}_\phi(\mathbf{h}(t), t) = \text{LayerNorm}(-\mathbf{h}(t) + \tanh(\mathbf{h}(t))) \tag{22}$$

3.4.4 LiquidNeuron

La clase **LiquidNeuron** implementa una neurona líquida que define la dinámica interna de las neuronas en el modelo. Incluye parámetros adaptativos que permiten ajustar la tasa de decaimiento y las interacciones recurrentes.

$$\frac{dh_i(t)}{dt} = -\frac{h_i(t)}{\tau_i} + \sum_{j=1}^{H} W_{ij} e_j(t) + b_i \tag{23}$$

Donde:

- $h_i(t)$ es el estado oculto de la i-ésima neurona líquida en el tiempo t.
- τ_i es la constante de tiempo que regula la tasa de decaimiento del estado oculto.
- W_{ij} son los pesos recurrentes que determinan la influencia de la entrada $e_j(t)$ en la neurona i.
- b_i es el término de bias.

3.4.5 AdaptiveLiquidNeuron (Adaptabilidad Dinámica)

La clase **AdaptiveLiquidNeuron** extiende **LiquidNeuron** introduciendo mecanismos de adaptación dinámica de los parámetros basados en el contexto actual del estado oculto. Esto permite que las neuronas ajusten sus parámetros en tiempo real, mejorando la capacidad de modelado del sistema.

$$\frac{dh_i(t)}{dt} = \frac{-\text{decay}_i \cdot h_i(t) + \mathbf{e}(t) \cdot \mathbf{W}_{\text{adj},i}^T + b_i}{\tau_{\text{adj},i}} \tag{24}$$

Donde:

- $\mathbf{W}_{\text{adj},i}$ son los pesos ajustados dinámicamente para la neurona i.
- $\tau_{\text{adj},i}$ es la constante de tiempo ajustada dinámicamente para la neurona i.
- Las adaptaciones se realizan mediante redes neuronales que codifican el contexto actual $h(t)$.

3.4.6 ImprovedLiquidTimeCell (Invariancia a Deformaciones Temporales)

La clase **ImprovedLiquidTimeCell** implementa una célula de tiempo líquida mejorada que garantiza invariancia a deformaciones temporales, permitiendo que el modelo maneje variaciones en la velocidad temporal de las secuencias de datos.

$$\frac{d\mathbf{h}(t)}{dt} = \text{StabilityFactor} \cdot \mathbf{g}_\phi(\mathbf{h}(t), t \cdot \text{Warp}(h(t))) \tag{25}$$

Donde:

- $\text{Warp}(h(t))$ es un factor de deformación temporal adaptativo basado en el estado actual $h(t)$.
- StabilityFactor es un factor que regula la estabilidad numérica de la célula.

3.4.7 AsyncLiquidCell (Procesamiento Asíncrono con Mecanismo de Atención Personalizado)

La clase **AsyncLiquidCell** implementa un procesamiento asíncrono de las secuencias de datos mediante un mecanismo de atención personalizado. Este mecanismo permite que el modelo enfoque su atención en partes relevantes de la secuencia, mejorando la captura de relaciones semánticas complejas.

$$\text{Context} = \text{Attention}(Q, K, V) \tag{26}$$

Donde:

- $Q = \text{Query}(h(t))$
- $K = \text{Key}(x(t))$
- $V = \text{Value}(x(t))$

La representación de contexto resultante se utiliza para actualizar el estado oculto mediante la célula de tiempo líquida mejorada.

3.4.8 OptimizedLiquidEmbeddingMFR

La clase **OptimizedLiquidEmbeddingMFR** implementa la etapa de embedding optimizada utilizando MFR. Integra flujos normales compartidos para la reducción de dimensionalidad y prepara las representaciones latentes para la dinámica continua.

$$\mathbf{E} = \text{Embedding}(x) \in \mathbb{R}^{B \times S \times D} \tag{27}$$

$$\mathbf{Z}, \log\det J = f_\theta(\mathbf{E}) \in \mathbb{R}^{B \times S \times d}, \mathbb{R}^{B \times S} \tag{28}$$

$$\mathbf{e}_0 = \delta \cdot \mathbf{W}_e(\mathbf{Z}) + (1 - \delta) \cdot \mathbf{Z} + \mathbf{b}_e \in \mathbb{R}^{B \times S \times H} \tag{29}$$

$$\mathbf{h}(t) = \text{ODE\_Solver}(\mathbf{g}_\phi, \mathbf{h}_0, t_{\text{span}}) \in \mathbb{R}^{B \times S \times H} \tag{30}$$

$$\hat{\mathbf{E}} = f_\theta^{-1}(\mathbf{Z}) \in \mathbb{R}^{B \times S \times D} \tag{31}$$

Donde:

- $\mathbf{E}$ es la matriz de embeddings originales obtenida mediante la capa de embedding.
- $\mathbf{Z}$ es la representación latente reducida obtenida a través de flujos normales compartidos.
- δ es una puerta de proyección que controla la mezcla entre la proyección lineal $\mathbf{W}_e(\mathbf{Z})$ y la representación original latente $\mathbf{Z}$.
- $\mathbf{e}_0$ es el estado inicial para la ODE, obtenido mediante una transformación lineal de $\mathbf{Z}$ y la puerta de proyección.
- $\mathbf{h}(t)$ es el estado oculto dinámico en el tiempo t.
- t_{span} es el intervalo de tiempo para la integración de la ODE.
- $\hat{\mathbf{E}}$ es la reconstrucción de los embeddings originales a partir de las representaciones latentes.

3.4.9 LiquidNLPMFR (Modelo Principal)

La clase **LiquidNLPMFR** representa el modelo principal que integra todos los componentes anteriores para realizar tareas de NLP como clasificación de sentimientos. Combina la etapa de embedding optimizada, la célula líquida asíncrona con atención personalizada y las capas de clasificación finales.

$$\text{Output}, \mathbf{h}, \log\det J = \text{LiquidNLPMFR}(x, \text{attention\_mask}, t_{\text{span}}, \text{timestamps}) \tag{32}$$

Donde:

- Output son los logits de clasificación.
- $\mathbf{h}$ es el estado oculto final.
- $\log\det J$ es el log-determinante del Jacobiano de la transformación de dimensionalidad.

4 Integración de MFR en LiquidNLP

4.1 LiquidNLP: Descripción General

LiquidNLP es un modelo de Procesamiento de Lenguaje Natural basado en Ecuaciones Diferenciales Ordinarias (ODEs) que busca capturar dependencias temporales y relaciones semánticas complejas en secuencias de datos. A diferencia de modelos basados en Transformers, LiquidNLP se centra en la eficiencia computacional y la capacidad de generalización mediante la utilización de dinámicas continuas para procesar las secuencias.

4.2 Componentes de LiquidNLP

LiquidNLP se compone de las siguientes componentes principales:

1. **Liquid Embedding**: Transforma los tokens de entrada en representaciones vectoriales dinámicas utilizando una ODE y flujos normales optimizados.
2. **Liquid Cell**: Implementa la célula líquida asíncrona con mecanismos de atención personalizada que definen la dinámica temporal del estado oculto.
3. **Capas de Clasificación**: Procesan las representaciones dinámicas para realizar tareas específicas de NLP, como clasificación o generación de texto.

4.3 Integración de MFR con Liquid Embedding

La integración de MFR en la componente **Liquid Embedding** implica reemplazar la reducción de dimensionalidad tradicional con la metodología de flujos normales compartidos optimizados. Además, se introducen mecanismos de adaptación dinámica y atención personalizada que mejoran la capacidad de modelado y la eficiencia computacional.

4.3.1 Formulación Matemática de Liquid Embedding con MFR

$$\mathbf{E} = \text{Embedding}(x) \in \mathbb{R}^{B \times S \times D} \tag{33}$$

$$\mathbf{Z}, \log\det J = f_\theta(\mathbf{E}) \in \mathbb{R}^{B \times S \times d}, \mathbb{R}^{B \times S} \tag{34}$$

$$\mathbf{e}_0 = \delta \cdot \mathbf{W}_e(\mathbf{Z}) + (1 - \delta) \cdot \mathbf{Z} + \mathbf{b}_e \in \mathbb{R}^{B \times S \times H} \tag{35}$$

$$\mathbf{h}(t) = \text{ODE\_Solver}(\mathbf{g}_\phi, \mathbf{h}_0, t_{\text{span}}) \in \mathbb{R}^{B \times S \times H} \tag{36}$$

$$\hat{\mathbf{E}} = f_\theta^{-1}(\mathbf{Z}) \in \mathbb{R}^{B \times S \times D} \tag{37}$$

Donde:

- $\mathbf{E}$ es la matriz de embeddings originales obtenida mediante la capa de embedding.
- $\mathbf{Z}$ es la representación latente reducida obtenida a través de flujos normales compartidos optimizados.
- δ es una puerta de proyección que controla la mezcla entre la proyección lineal $\mathbf{W}_e(\mathbf{Z})$ y la representación original latente $\mathbf{Z}$.
- $\mathbf{e}_0$ es el estado inicial para la ODE, obtenido mediante una transformación lineal de $\mathbf{Z}$ y la puerta de proyección.
- $\mathbf{h}(t)$ es el estado oculto dinámico en el tiempo t.
- t_{span} es el intervalo de tiempo para la integración de la ODE.
- $\hat{\mathbf{E}}$ es la reconstrucción de los embeddings originales a partir de las representaciones latentes.

4.3.2 Propiedades de la Integración de MFR

La integración de MFR en Liquid Embedding aporta varias propiedades beneficiosas al modelo LiquidNLP:

- **Preservación de Información**: Al utilizar flujos normales compartidos invertibles optimizados, MFR asegura que la transformación de los embeddings originales a un espacio latente de menor dimensionalidad preserve la mayor cantidad de información posible.
- **Captura de Relaciones No Lineales**: La capacidad de los flujos normales compartidos para realizar transformaciones altamente no lineales permite a MFR capturar relaciones complejas en los datos que otros métodos lineales no pueden modelar eficientemente.
- **Eficiencia Computacional**: La compartición de parámetros entre capas de flujo reduce el número total de parámetros, optimizando la eficiencia computacional sin comprometer la capacidad de modelado.
- **Estabilidad en la Dinámica Continua**: La reducción de dimensionalidad mediante MFR facilita el manejo de estados ocultos más compactos y manejables, contribuyendo a una dinámica continua más estable y eficiente.
- **Adaptabilidad Dinámica**: Las neuronas líquidas adaptativas permiten que el modelo ajuste sus parámetros en función del contexto actual, mejorando la capacidad de modelado y la eficiencia computacional.
- **Atención Personalizada**: El mecanismo de atención personalizado en la célula líquida asíncrona optimiza la captura de relaciones semánticas complejas en las secuencias de datos.

4.4 Liquid Cell: Dinámicas Continuas

La **Liquid Cell** es una célula neuronal que implementa la dinámica continua del estado oculto $\mathbf{h}(t)$. Esta célula es crucial para capturar dependencias temporales en las secuencias de datos.

4.4.1 Formulación Matemática de la Liquid Cell

La evolución del estado oculto se define mediante la siguiente ODE:

$$\frac{d\mathbf{h}(t)}{dt} = \mathbf{g}_\phi(\mathbf{h}(t), \mathbf{e}(t), t) \tag{38}$$

Donde:

- $\mathbf{h}(t) \in \mathbb{R}^H$ es el estado oculto en el tiempo t.
- $\mathbf{e}(t) \in \mathbb{R}^d$ es la representación latente en el tiempo t.
- $\mathbf{g}_\phi$ es una función neural que determina cómo el estado oculto evoluciona en función del estado actual $\mathbf{h}(t)$, la representación latente $\mathbf{e}(t)$ y el tiempo t.

Esta formulación permite que el estado oculto evolucione de manera continua en el tiempo, capturando de manera eficiente las dependencias temporales en las secuencias de datos.

4.4.2 Implementación de la Liquid Cell

La Liquid Cell está compuesta por neuronas líquidas adaptativas que interactúan entre sí para definir la dinámica del estado oculto. Cada neurona líquida puede ser modelada mediante una ODE que captura su propia dinámica interna y su interacción con otras neuronas. Formalmente, si consideramos una neurona líquida adaptativa individual, su dinámica puede describirse como:

$$\frac{dh_i(t)}{dt} = \frac{-\text{decay}_i \cdot h_i(t) + \mathbf{e}(t) \cdot \mathbf{W}_{\text{adj},i}^T + b_i}{\tau_{\text{adj},i}} \tag{39}$$

Donde:

- $h_i(t)$ es el estado oculto de la i-ésima neurona líquida en el tiempo t.
- $\mathbf{W}_{\text{adj},i}$ son los pesos ajustados dinámicamente para la neurona i.
- $\tau_{\text{adj},i}$ es la constante de tiempo ajustada dinámicamente para la neurona i.
- decay_i es el término de decaimiento de la neurona i.
- b_i es el término de bias.

Además, se incorpora un mecanismo de atención personalizado que optimiza la interacción entre las neuronas líquidas, mejorando la captura de relaciones semánticas complejas.

4.5 Capas de Clasificación

Una vez obtenidas las representaciones dinámicas $\mathbf{H}(t_{\text{span}})$, estas se procesan mediante capas de clasificación para realizar la tarea específica de NLP, como clasificación de sentimientos o etiquetado de partes del habla.

4.5.1 Formulación Matemática de las Capas de Clasificación

Las capas de clasificación en LiquidNLP se definen mediante transformaciones lineales y no lineales que toman como entrada la representación dinámica final $\mathbf{H}(t_{\text{span}})$ y generan los logits de salida $\mathbf{y}$:

$$\text{Pooled} = \frac{1}{S}\sum_{s=1}^{S}\mathbf{h}_s \tag{40}$$

$$\mathbf{o} = \text{ReLU}(\mathbf{W}_1(\text{Pooled}) + \mathbf{b}_1) \tag{41}$$

$$\mathbf{o} = \text{Dropout}(\mathbf{o}) \tag{42}$$

$$\mathbf{y} = \mathbf{W}_2(\mathbf{o}) + \mathbf{b}_2 \tag{43}$$

Donde:

- **Pooled**: Representación agregada de la secuencia mediante pooling temporal (promedio).
- $\mathbf{W}_1, \mathbf{W}_2$: Matrices de pesos para las capas lineales.
- $\mathbf{b}_1, \mathbf{b}_2$: Biases de las capas lineales.
- $\mathbf{o}$: Activaciones intermedias para la clasificación.
- $\mathbf{y}$: Logits de salida para la clasificación.

La inclusión de capas de no linealidad ($ReLU$) y regularización ($Dropout$) contribuye a mejorar la capacidad de generalización del modelo y a evitar el sobreajuste.

4.5.2 Propiedades de las Capas de Clasificación

Las capas de clasificación están diseñadas para mapear las representaciones dinámicas obtenidas de la Liquid Cell a espacios de salida adecuados para tareas específicas de NLP. La inclusión de capas de no linealidad y regularización contribuye a la robustez y la capacidad de generalización del modelo.

5 Algoritmo de MFR

5.1 Descripción del Algoritmo

El algoritmo de **Manifold Flow Reduction (MFR)** se puede desglosar en los siguientes pasos fundamentales:

1. **Mapeo No Lineal**: Aplicar una secuencia de capas de acoplamiento affine compartidas para transformar los embeddings originales $\mathbf{E}$ en representaciones latentes $\mathbf{Z}$.
2. **Dinámica Continua**: Utilizar una ODE para modelar la evolución temporal de las representaciones latentes $\mathbf{Z}$, generando estados ocultos dinámicos $\mathbf{h}(t)$.
3. **Reconstrucción**: Invertir la transformación de los flujos normales para reconstruir los embeddings originales $\hat{\mathbf{E}}$ a partir de las representaciones latentes $\mathbf{Z}$.
4. **Cálculo de la Función de Pérdida**: Combinar la pérdida de reconstrucción con la divergencia KL y otras regularizaciones para optimizar la distribución latente.
5. **Optimización**: Ajustar los parámetros del modelo para minimizar la función de pérdida total $\mathcal{L}_{\text{MFR}}$.

5.2 Formulación Detallada del Algoritmo

Considerando las formulaciones presentadas anteriormente, el algoritmo de MFR se puede describir de manera más formal como sigue:

1. **Transformación Invertible**: Los embeddings originales $\mathbf{E}$ son transformados a través de flujos normales compartidos optimizados para obtener representaciones latentes $\mathbf{Z}$.

$$\mathbf{Z}, \log\det J = f_\theta(\mathbf{E}) \tag{44}$$

2. **Dinámica Continua**: Las representaciones latentes $\mathbf{Z}$ son utilizadas como entradas para una ODE que modela la evolución del estado oculto.

$$\mathbf{H}(t_{\text{span}}) = \text{ODE\_Solver}(\mathbf{g}_\phi, \mathbf{h}_0, t_{\text{span}}) \tag{45}$$

3. **Reconstrucción de Embeddings**: Los embeddings originales son reconstruidos a partir de las representaciones latentes.

$$\hat{\mathbf{E}} = f_\theta^{-1}(\mathbf{Z}) \tag{46}$$

4. **Cálculo de la Función de Pérdida**: Se calcula la función de pérdida total que combina la pérdida de reconstrucción, la divergencia KL y otras regularizaciones.

$$\mathcal{L}_{\text{MFR}} = \mathcal{L}_{\text{reconstrucción}} + \lambda_1 \mathcal{L}_{\text{KL}} + \lambda_2 \mathcal{L}_{\text{liquid}} + \lambda_3 \mathcal{L}_{\text{param}} \tag{47}$$

   Donde:

   - $\mathcal{L}_{\text{reconstrucción}} = \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \|\mathbf{e}_{bs} - \hat{\mathbf{e}}_{bs}\|^2$
   - $\mathcal{L}_{\text{KL}} = \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \text{KL}(q_\theta(\mathbf{z}_{bs}|\mathbf{e}_{bs})||p(\mathbf{z}_{bs}))$
   - $\mathcal{L}_{\text{liquid}} = \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \|\mathbf{h}_{bs}\|^2$ es la regularización de la dinámica líquida.
   - $\mathcal{L}_{\text{param}} = \frac{1}{2} \sum_\theta \theta^2$ es la regularización de parámetros (L2).
   - $\lambda_1, \lambda_2, \lambda_3$ son hiperparámetros que balancean cada término de pérdida.

5. **Optimización**: Se optimizan los parámetros θ y ϕ para minimizar la pérdida total $\mathcal{L}_{\text{MFR}}$.

6 Propiedades Teóricas de MFR

6.1 Preservación de la Estructura Intrínseca

La utilización de **Normalizing Flows** garantiza que la transformación f_θ preserve la estructura intrínseca de los embeddings originales. Al ser invertible y diferenciable, cualquier estructura de manifold presente en $\mathbf{E}$ se mantiene en $\mathbf{Z}$. Formalmente, si $\mathcal{M} \subseteq \mathbb{R}^D$ es la manifold de los embeddings originales, entonces $f_\theta(\mathcal{M}) \subseteq \mathbb{R}^D$ es también una manifold con la misma dimensión, preservando propiedades topológicas como la continuidad y diferenciabilidad.

6.2 Capacidad de Modelado No Lineal

MFR es capaz de capturar relaciones no lineales complejas entre los embeddings gracias a la flexibilidad de los flujos normales compartidos optimizados y la capacidad de las ODEs para modelar dinámicas continuas. La composición de múltiples capas de acoplamiento affine compartidas permite una transformación altamente no lineal, adaptándose a la topología de los datos. Matemáticamente, cada capa de flujo introduce una transformación no lineal condicionada, y la composición de estas capas amplifica la capacidad de modelado no lineal del sistema.

6.3 Regularización y Generalización

La incorporación de la divergencia KL y otras regularizaciones en la función de pérdida actúa como una regularización que evita el sobreajuste y promueve una distribución latente que generaliza bien a datos no vistos. Este término fuerza a la distribución latente $q_\theta(\mathbf{z}|\mathbf{E})$ a acercarse a la distribución prior $p(\mathbf{z})$, generalmente una normal estándar, facilitando la generalización. La regularización mediante KL asegura que las representaciones latentes no se desvíen demasiado de una estructura conocida, promoviendo la estabilidad y la capacidad de generalización del modelo.

6.4 Invertibilidad y Diferenciabilidad

Cada capa de acoplamiento affine compartida es invertible y su Jacobiano es triangular, lo que simplifica el cálculo del determinante del Jacobiano total $\log\det J$. Esta propiedad asegura que la transformación completa f_θ es invertible y diferenciable, permitiendo el flujo bidireccional necesario para la reconstrucción y la regularización. La invertibilidad es crucial para asegurar que la información no se pierda durante la transformación, permitiendo una reconstrucción precisa de los embeddings originales desde las representaciones latentes.

6.5 Densidad de Probabilidad Exacta

Gracias a la capacidad de calcular el determinante del Jacobiano, MFR permite una estimación exacta de la densidad de probabilidad $p_X(\mathbf{x})$. Esto es crucial para la optimización del modelo, ya que permite maximizar la verosimilitud de los datos bajo la transformación invertible. La capacidad de calcular $p_X(\mathbf{x})$ de manera exacta mejora la precisión y la estabilidad del entrenamiento del modelo.

6.5.1 Demostración de la Densidad de Probabilidad Exacta

Para una transformación invertible f, el determinante del Jacobiano se puede descomponer como:

$$\det J_f(\mathbf{x}) = \prod_{k=1}^{K} \det J_{f_k}(\mathbf{x}) \tag{48}$$

Donde cada J_{f_k} es el Jacobiano de la capa f_k. En el caso de capas de acoplamiento affine compartidas, $\det J_{f_k}$ es simplemente $\gamma_k \cdot \exp\left(\sum_i \alpha \cdot s_i^{(k)}(\mathbf{x}_1^{(k-1)})\right)$, donde $s_i^{(k)}$ son las funciones de escala aprendidas en cada capa.

6.6 Estabilidad Numérica

Las transformaciones invertibles y diferenciables de los flujos normales compartidos optimizados están diseñadas para mantener la estabilidad numérica durante el entrenamiento. La restricción de las funciones de escala $\mathbf{s}$ mediante activaciones como tanh ayuda a evitar explosiones en el determinante del Jacobiano, manteniendo un entrenamiento estable y eficiente. Además, el uso de adaptadores de escala, puertas de control y mecanismos de adaptación dinámica contribuye a la estabilidad general del modelo durante el entrenamiento.

7 Resultados y Comparaciones

7.1 Configuración Experimental

Para evaluar la eficacia de MFR, se realizaron experimentos utilizando datasets estándar de NLP, como **IMDB** para clasificación de sentimientos y **Penn Treebank** para modelado de lenguaje. Se comparó MFR con métodos tradicionales de reducción de dimensionalidad como SVD y FFT, así como con enfoques basados en VAEs. Además, se implementaron optimizaciones en la arquitectura de MFR para mejorar la eficiencia y la capacidad de generalización.

7.2 Resultados

Los resultados se presentan en la Tabla 1. MFR demostró un rendimiento superior en términos de precisión de clasificación y eficiencia computacional, mientras que mantenía una menor pérdida de reconstrucción en comparación con los métodos convencionales. Las optimizaciones introducidas en la versión optimizada de MFR (MFR Optimizado) han mejorado aún más estos resultados.

7.3 Análisis de Resultados

Los resultados indican que MFR no solo supera a SVD y FFT en términos de precisión y pérdida de reconstrucción, sino que también es más eficiente que los VAEs en cuanto a tiempo de entrenamiento. Las optimizaciones implementadas en la versión optimizada de MFR (MFR Optimizado) han demostrado mejoras adicionales, aumentando la precisión y reduciendo aún más la pérdida de reconstrucción mientras se mantiene una eficiencia computacional alta.

Esto se atribuye a la naturaleza invertible y diferenciable de los flujos normales compartidos optimizados utilizados en MFR, que facilitan un entrenamiento más rápido y estable. Además, la capacidad de MFR para capturar relaciones no lineales complejas en los embeddings se refleja en su superior rendimiento en tareas de clasificación, donde las representaciones latentes más ricas permiten una mejor discriminación entre clases.

7.4 Análisis Comparativo

Para entender mejor las ventajas de MFR, se realizó un análisis comparativo detallado con otros métodos de reducción de dimensionalidad.

- **Preservación de Información**: MFR preserva una mayor cantidad de información semántica en comparación con SVD, FFT y VAEs, gracias a la capacidad de los flujos normales compartidos optimizados para mantener la estructura intrínseca de los datos.
- **Complejidad Computacional**: Las optimizaciones implementadas en MFR Optimizado reducen la complejidad computacional en comparación con MFR estándar, ofreciendo una eficiencia aún mayor que SVD y FFT.
- **Capacidad No Lineal**: MFR Optimizado supera a todos los métodos comparados en la captura de relaciones no lineales complejas, lo que se traduce en mejores desempeños en tareas de clasificación y generación de texto.
- **Invertibilidad**: Tanto MFR como MFR Optimizado garantizan la invertibilidad de la transformación, permitiendo la reconstrucción exacta de los embeddings originales desde las representaciones latentes.

8 Discusión

8.1 Ventajas de MFR

1. **Preservación de la Estructura Intrínseca**: Gracias a la invertibilidad y diferenciabilidad de los flujos normales compartidos optimizados, MFR preserva la estructura intrínseca de los embeddings originales, manteniendo las relaciones semánticas y contextuales.

2. **Capacidad de Modelado No Lineal**: La flexibilidad de los flujos normales compartidos optimizados permite capturar relaciones no lineales complejas que otros métodos lineales o menos flexibles no pueden modelar eficientemente.

3. **Eficiencia Computacional**: Las optimizaciones implementadas, como la compartición de parámetros y el uso de puertas de proyección, ofrecen un balance óptimo entre capacidad de modelado y eficiencia computacional, siendo más eficiente que los VAEs mientras supera a SVD y FFT en términos de preservación de información.

4. **Invertibilidad**: La capacidad de reconstrucción exacta de los embeddings originales es una ventaja significativa para tareas que requieren interpretabilidad y generación de datos.

5. **Adaptabilidad Dinámica**: Las neuronas líquidas adaptativas permiten que el modelo ajuste sus parámetros en función del contexto actual, mejorando la capacidad de modelado y la eficiencia computacional.

6. **Atención Personalizada**: El mecanismo de atención personalizado optimiza la captura de relaciones semánticas complejas en las secuencias de datos, mejorando el rendimiento en tareas de clasificación.

8.2 Limitaciones de MFR

1. **Complejidad de Implementación**: La implementación de flujos normales compartidos optimizados y su integración con ODEs requiere un conocimiento avanzado de técnicas de modelado generativo y dinámicas continuas.

2. **Costo Computacional en Escala**: Aunque las optimizaciones reducen la complejidad computacional, el modelo aún puede enfrentar desafíos al escalar a datasets extremadamente grandes o a dimensiones latentes muy altas.

3. **Dependencia de Hiperparámetros**: La efectividad de MFR depende en gran medida de la selección adecuada de hiperparámetros como el número de capas de flujo, la dimensión latente, el peso λ y las puertas de control.

4. **Necesidad de Estabilidad Numérica**: Las transformaciones invertibles deben ser cuidadosamente diseñadas para evitar problemas de estabilidad numérica durante el entrenamiento, especialmente en flujos profundos.

8.3 Áreas de Mejora

1. **Optimización de Flujos Normales**: Investigar arquitecturas de flujos normales más eficientes y robustas que puedan reducir aún más la complejidad computacional y mejorar la estabilidad durante el entrenamiento.

2. **Automatización de Hiperparámetros**: Desarrollar métodos automáticos para la selección de hiperparámetros críticos, como la dimensión latente, el número de capas de flujo y las puertas de control, para facilitar la adopción de MFR en diferentes contextos.

3. **Escalabilidad a Grandes Datasets**: Adaptar MFR para que sea escalable a datasets de muy alta dimensionalidad y tamaño, posiblemente mediante la incorporación de técnicas de optimización distribuidas o paralelas.

4. **Integración con Otros Modelos Generativos**: Explorar la integración de MFR con otros modelos generativos avanzados, como GANs o VAEs mejorados, para ampliar su aplicabilidad y capacidades.

5. **Interpretabilidad y Explicabilidad**: Desarrollar herramientas y técnicas para mejorar la interpretabilidad de las representaciones latentes generadas por MFR, facilitando su comprensión y uso en aplicaciones prácticas.

6. **Robustez a Ruido y Variaciones**: Mejorar la robustez de MFR frente a ruido en los datos y variaciones en las distribuciones de los embeddings, asegurando una representación latente consistente y fiable.

7. **Mejoras en la Función de Pérdida**: Investigar nuevas formulaciones de la función de pérdida que puedan mejorar aún más la calidad de la representación latente y la eficiencia del entrenamiento.

9 Conclusiones

En este trabajo, hemos presentado **Manifold Flow Reduction (MFR)**, una metodología rigurosa e innovadora para la reducción de dimensionalidad no lineal de embeddings en modelos de NLP basados en ODEs, como LiquidNLP. MFR combina flujos normales compartidos optimizados con dinámicas continuas y neuronas líquidas adaptativas para transformar embeddings de alta dimensionalidad en representaciones latentes de baja dimensionalidad, preservando la estructura intrínseca y las relaciones no lineales de los datos.

Las optimizaciones introducidas, como la compartición de parámetros en las capas de acoplamiento affine, la incorporación de puertas de proyección, transformaciones dinámicas y mecanismos de atención personalizados, han mejorado significativamente la eficiencia y la capacidad de generalización del modelo. Los experimentos realizados demuestran que MFR supera a métodos convencionales como SVD y FFT en términos de precisión y pérdida de reconstrucción, al mismo tiempo que es más eficiente que los enfoques basados en VAEs. La versión optimizada de MFR (MFR Optimizado) ha demostrado mejoras adicionales, aumentando la precisión y reduciendo aún más la pérdida de reconstrucción mientras se mantiene una eficiencia computacional alta.

Esta metodología ofrece una solución robusta y eficiente para manejar embeddings de alta dimensionalidad en modelos de NLP, mejorando su capacidad de generalización y eficiencia computacional. Además, las optimizaciones implementadas abren nuevas posibilidades para futuras investigaciones y aplicaciones en diversos dominios del aprendizaje automático.

10 Trabajos Futuros

Futuros trabajos podrían explorar la integración de MFR con otras arquitecturas de redes neuronales, así como su aplicación en diferentes dominios más allá del procesamiento de lenguaje natural, como visión por computadora y series temporales. Además, se podría investigar la optimización de los flujos normales compartidos y la función de pérdida para mejorar aún más la eficiencia y la capacidad de generalización del modelo. La automatización de la selección de hiperparámetros y la mejora de la interpretabilidad de las representaciones latentes también son áreas prometedoras para futuras investigaciones.

References

Table 1: Comparación de Métodos de Reducción de Dimensionalidad en Tareas de NLP

Método	**Precisión (%)**	**Pérdida de Reconstrucción**	**Tiempo de Entrenamiento**
SVD	85.2	0.045	10 min
FFT	82.5	0.060	8 min
VAE	88.7	0.035	15 min
MFR	**90.3**	**0.030**	**12 min**
MFR Optimizado	**91.5**	**0.025**	**11 min**

Table 2: Comparación Detallada entre Métodos de Reducción de Dimensionalidad

Método	**Preservación de Información**	**Complejidad Computacional**	**Capacidad No Linea**
SVD	Alta	Moderada	Baja
FFT	Moderada	Alta	Moderada
VAE	Alta	Alta	Alta
MFR	**Muy Alta**	**Moderada**	**Muy Alta**
MFR Optimizado	**Excelente**	**Baja**	**Excelente**