**Propuesta**

Consta de dos fases

1. **Fase 1:** Preentrenamiento

El objetivo de esta fase es generar un código binario o prehash para cada elemento del conjunto de entrenamiento, el cual será utilizado más adelante durante la fase 2 para entrenar a la red neuronal. Es debido a lo anterior que los códigos creados durante esta fase deben preservar en la medida de lo posible la distribución de los elementos en el espacio original, a esta restricción se le conoce como *locality preserving constraint.* Al igual que se hizo en [Convolutional Neural Networks for Text Hashing] la técnica utilizada para calcular los prehash es una variante del Spectral Hashing propuesto en [Spectral Hashing] el cual cumple con la restricción anteriormente descrita.

En [Spectral Hashing] Weiss y su equipo enumeran los requisitos que todo buen hash debe tener cumplir:

* 1. Ser fácil de calcular para nuevos elementos.
  2. Ser compacto.
  3. Cumplir con la *locality preserving constraint*

Ignorando la primera restricción es posible plantear la elección de los códigos binarios como un problema de optimización. Sea $B$ la matrix cuyas filas corresponden a los códigos binarios de dimensión $k$ asociados a los $n$ elementos del conjunto de entrenamiento siendo $b\_i = B\_{i,:}^T$ y sea $W\_{n \times n}$ la matriz de afinidad asociada a dichos $n$ elementos, se plantea el siguiente problema de optimización:

\arg\min\_{B} \sum\_{i,j=1}^{n} W\_{ij} \left \| b\_i - b\_j \right \|^2

Sujeto a:

b\_i \in \left \{ -1, 1 \right \}^k

\sum\_{i=1}^{n} b\_i= 0

\frac{1}{n}\sum\_{i=1}^{n} b\_ib\_i^T = I

En donde la restricción \sum\_{i=1}^{n} b\_i= 0 mantiene el balance entre los bits al forzar una distribución uniforme entre las columnas de la matriz, mientras que la restricción \frac{1}{n}\sum\_{i=1}^{n} b\_ib\_i^T = I asegura que los bits no estén correlacionados.

Como se demuestra en [Spectral Hashing], el problema anterior es equivalente a un problema de partición de grafos. Específicamente a encontrar los cortes mínimos que permiten obtener k particiones balanceadas para el grafo completo cuyos nodos son los $n$ elementos de la colección y el peso de sus aristas esta dado por W\_{i,j}. Este problema es NP-Hard por lo que para resolverlo se utiliza la matrix laplaciana $L$ del grafo, la cual se define como $L = D – W$ siendo $D$ una matriz diagonal de $n \times n$ en donde D\_{i,i} = \sum\_{j=1}^{n} W\_{i,j}, y el método de relajación espectral, por lo que el problema anterior se puede reescribir como:

\arg\min\_{B} tr(B^T(D-W)B)

Sujeto a:

B^T1=0

B^TB=I

Si se excluye la restricción $b\_i \in \left \{ -1, 1 \right \}^k$ la solución al problema anterior esta dada por los $k$ vectores propios de la matriz laplaciana $D-W$ con mínimos valores propios (excluyendo la solución trivial, es decir el vector con valor propio igual a 0) luego:

B = [v\_2 \;\; v\_3 \;\; \ldots \;\; v\_{k+1}]

Donde v\_2 denota el vector propio cuyo valor propio es el segundo valor positivo más pequeño, de igual forma con el tercero y así sucesivamente.

Finalmente, para determinar los prehash del conjunto de entrenamiento se aplica un funcion umbral para obtener de esta forma una cadena binaria.

Ahora bien, dado que los prehash se genera en base a los vectores propios de $\bm{L}$ cobra especial relevancia el método utilizado para construir $\bm{W}$ ya que la matriz laplaciana surge a partir de esta última. Para efectos de esta memoria se han abordado 3 métodos para construir dicha matriz, siendo el primero de ellos el método original propuesto en \cite{SpectralHashing} mientras que los dos restantes corresponden a la aproximación descrita en \cite{ Convolutional Neural Networks for Text Hashing}.

W\_{i,j} = \exp{\left ( -\frac{\left \| x\_i - x\_j \right \|^2}{\epsilon^2} \right )}

La ecuación anterior corresponde al kernel gaussiano utilizado en \cite{SpectralHashing}. Las variables $x\_i$ y $x\_j$ son elementos del conjunto de entrenamiento representados como vectores en $\mathbb{R}^d$. Para obtener dicha representación vectorial se utiliza el vector documento de la matriz TF-IDF de la colección, de la misma forma en que fue utilizada en \cite{ ConvolutionalNeuralNetworksforTextHashing}. Por su parte la constate $\epsilon$ define la distancia en $\mathbb{R}^d$ a la que dos ítems pueden considerarse similares. Este tipo de kernel es ampliamente utilizado como medida de similitud debido a que tiende a 0 entre más disimiles sean $x\_i$ y $x\_j$ y a 1 entre más semejantes son.

W\_{i,j} =\left\{

\begin{matrix}

c\_{ij}\cdot\exp{\left ( -\frac{\left \| x\_i - x\_j \right \|^2}{2\sigma^2} \right )}, & \textrm{si} & x\_i\in \mathbf{N}\_k(x\_j) \; \lor \; x\_j\in \mathbf{N}\_k(x\_i)

\\

0, & & \textrm{en cualquier otro caso}

\end{matrix}\right.

La ecuación anterior corresponde a una de las variantes del *heat kernel* utilizado en \cite{ ConvolutionalNeuralNetworksforTextHashing}. Aunque a primera vista similar al kernel gaussiano presentado anteriormente, este introduce un coeficiente de ajuste $c\_ij$ y un parámetro $\sigma$ que cumple la funcion de $\epsilon$ del kernel anterior. No obstante, a diferencia de este ultimo la funcion solo es activada cuando uno de los elementos pertenece a los $k$ vecinos más cercanos del otro, lo que se denota mediante la funcion de vecindad $\mathbf{N}\_k(\dot)$.

Por ultimo en \cite{ ConvolutionalNeuralNetworksforTextHashing} se propone una variante al kernel anterior aprovechando la información proveniente de las etiquetas del dataset. Dicho lo anterior se establece un mayor valor para $c\_ij$ al que denominaremos como $a$ cuando ambos elementos $x\_i$ y $x\_j$ comparten una etiqueta y un valor menor $b$ cuando no lo hacen.

1. **Fase 2:** Entrenamiento y Predicción