**Implementación**

En este capítulo se presenta la implementación real de la propuesta señalada en el capítulo anterior junto a los resultados obtenidos.

**Datasets**

Para la realización de los experimento se emplearon los mismos conjuntos de datos utilizados en [Convolutional Neural Nerworks for Text hashing]; esto con la finalidad de poder comparar los resultados obtenidos con los resultados presentados en ese trabajo. Dicho lo anterior los dataset utilizados son 20Newsgroup y SearchSnippets, los cuales se detallan a continuación.

1. **20 Newsgroups:** Es una colección de datos de aproximadamente 20.000 entradas pertenecientes al repositorio newsgroup de la red Usenet. Estos documentos se encuentran distribuidos de manera aproximadamente uniforme a lo largo de 20 categorías diferentes, cada una perteneciente a un tópico en particular, siendo un dataset muy popular para tareas de clasificación y clustering de texto. [http://qwone.com/~jason/20Newsgroups/]. La versión del dataset utilizada para la elaboración de este trabajo es la variante bydate procesada por la doctora Ana Cardoso Cachopo [http://ana.cachopo.org/datasets-for-single-label-text-categorization], la cual realizo un proceso de steaming utilizando el algoritmo de porter. Además, se removieron de la colección aquellos documentos con mas de 300 términos.
2. **SearchSnippets:** Es un dataset de aproximadamente 13.000 elementos generado a partir de resultados de búsqueda en Google utilizando para ello frases predefinidas para 8 dominios diferentes [http://gibbslda.sourceforge.net/fp224-phan.pdf]. Esta compuesto de textos pequeños de una o dos oraciones, donde las stop words fueron removidas.

A continuación, se presenta una tabla con información estadística relevante de ambos Datasets.

Tabla : Información estadística de los datasets utilizados. C es el número de clases (etiquetas) correspondiente a cada dataset. Train/Test corresponde al numero de documentos pertenecientes al conjunto de entrenamiento (train) y al conjunto de prueba (test) respectivamente. L(mean/max) corresponde a la cantidad promedio y máxima de palabras por documento en el dataset. |V| representa la cardinalidad del vocabulario del dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **C** | **Train/Test** | **L(mean/max)** | **|V|** |
| 20 Newsgroups | 20 | 10443/6972 | 94.2/300 | 54370 |
| SearchSnippets | 8 | 10060/2280 | 18.1/50 | 29257 |

El objetivo de usar estos datasets es evaluar el desempeño del método implementado sobre colecciones de documentos de diferentes extensiones, cortos en el caso de SearchSnippets y de mayor extensión en 20 Newsgroups.

**Preentrenamiento**

Como fue mencionado en la sección homónima del apartado de propuesta, el objetivo de esta fase es generar un prehash para todos los elementos pertenecientes a los conjuntos de entrenamiento. Para lograr lo anterior se han utilizado las 3 variantes del spectral hashing descritas en aquella sección, las que pasaran a denominarse Weiss, Wang y Wang supervisado, haciendo referencia a los investigadores titulares de las respectivas publicaciones.

Dado lo anterior es necesario seleccionar los hiperparámetro correspondientes a cada método que permitan maximizar su poder de discriminación. Estos es seleccionar el parámetro $\epsilon$ en el caso de Weiss, el parámetro $\sigma$, el parámetro $k$ y el coeficiente de ajuste $c\_ij$ en el caso de Wang, agregando los valores de $a$ y $b$ para Wang supervisado. Por su parte también es necesario seleccionar la longitud de las cadenas a utilizar, por lo que se estudiara el comportamiento de los métodos a 16, 32 y 64 bits respectivamente.

Para determinar la calidad de prehashes generados se utilizó como métrica de evaluación mean precisión sobre el top 100 de documentos recuperados (mP@100) en relación con la distancia de hamming relativa entre estos y el documento de consulta. A su vez para realizar la búsqueda por similitud se realizó una partición aleatoria del conjunto de prehashes reservando el 20% como documentos de consulta. Este proceso de partición aleatoria y búsqueda fue realizado tres veces para minimizar los posibles sesgos provenientes de la selección del conjunto de validación. Los resultados presentados corresponden al promedio de los resultados del mP@100 sobre las tres particiones anteriormente descritas.

No obstante, antes de presentar los resultados obtenidos, es importante precisar como se construyeron los vectores tf-idf que representan a los elementos de la colección. Para realizar esto, se utilizó la función TfidfVectorizer de la librería Sklearn la cual calcula el coeficiente idf introduciendo una variación respecto a la formulación expuesta en la ecuación \ref{eqn:tf-idf} del capítulo antecedentes.

\begin{gather}

\text{idf}(t, D) = \log\left(\frac{\text{número total de documentos dentro de la colección\ } D}{\text{número de documentos en\ } D \text{\, que contienen el término\ } t}\right)+1\\[15pt]

\end{gather}

El propósito de sumar $1$ en idf radica en evitar que la técnica ignore a aquellos términos que, de lo contrario, tendrían un valor idf igual a cero, es decir a aquellos que aparecen en todos los documentos de la colección

\cite{<https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer>}.

Aclarado el punto anterior, se procede a presentar los resultados obtenidos divididos por método de preentrenamiento.

**Weiss**

Es el método original descrito en \cite{Spectral Hashing}. En el se utiliza un kernel gaussiano para generar la matriz $W$.

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

Como puede apreciarse la ecuación genera una matriz $W$ densa lo que facilita el calculo de los vectores y valores propios de la matriz laplaciana. Este hecho provoca que el tiempo necesario para calcular los prehashes utilizando este método sea considerablemente menor al tiempo empleado por Wang y Wang Supervisado. Lo anterior sumado a que el método posee únicamente dos parámetros, uno explicito con $\epsilon$ y uno implícito con número de bits, permite realizar un proceso de sintonización mas exhaustivo que el realizado para los otros dos métodos.

A continuación, se presenta los resultados obtenidos al variar el parámetro $\epsilon$ utilizando cadenas de 16, 32 y 64 bits sobre los conjuntos 20 Newsgroups y SearchSnippets respectivamente.

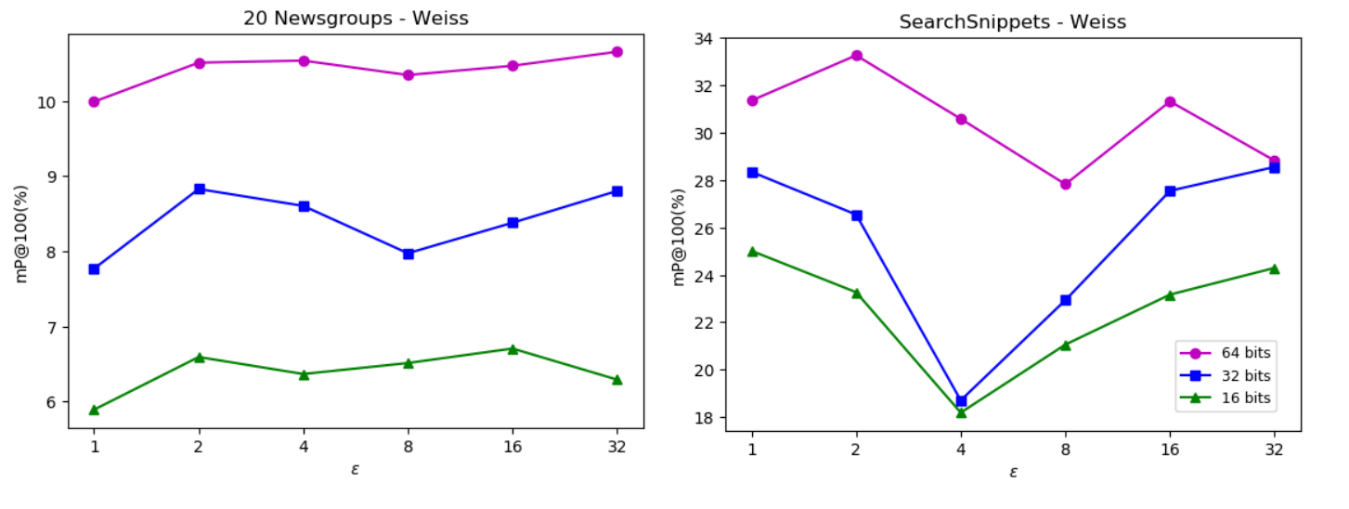


Ilustración :Rresultados de mP@100 en funcion del parámetro épsilon utilizando cadenas de 16, 32 y 64 bits. Resultados sobre 20 Newsgroups a la izquierda y SearchSnippets a la derecha.

################################################################################

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

La ecuación anterior corresponde al kernel gaussiano

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}.

Para evaluar la calidad de los prehashes generados, estos fueron particionados dejando un 20% para realizar una búsqueda por similitud sobre el 80% restante.