# PARCIALITO DE HASHING Y LSH

Franco Nicolás Batastini

103775

## *Consigna*

**Hashing y LSH 2020-2C**

Contamos con una colección de 1300 millones de tweets con fake news. Queremos usarlos para detectar otros tweets que puedan contener mensajes parecidos para poder filtrarlos.

El objetivo final de nuestra aplicación es dado un tweet buscar cuáles son tweets parecidos en nuestra base de 1300 millones de tweets usando LSH para luego decidir si el tweet es o no un fake news.Si hay tweets muy similares al nuevo en nuestra base de datos, lo categorizamos como fake news; caso contrario, no. El criterio que fijamos es que si el tweet actual tiene una semejanza mayor o igual a 0.73 con alguno de nuestra base de fake news entonces es un fake news.

Se decide usar la distancia de Jaccard como métrica para la construcción de nuestro esquema de LSH. Queremos que si la semejanza entre tweets es mayor o igual a 0.55 tengamos más de un 70% de probabilidad de que sean candidatos. Por otro lado, si la semejanza es menor o igual a 0.05 queremos tener menos de un 1% de probabilidad de que sean candidatos a ser comparados.

En base a esta información le pedimos que responda las siguientes preguntas:

1. ¿Cuántos minhash hacen falta y con qué tipo de esquema (b y r) ? Detalle los cálculos realizados y estime cantidad de falsos positivos y falsos negativos que vamos a tener sobre una base de 10000 tweets nuevos. (30 puntos)

2. Describa la etapa de pre-procesamiento en la cual tiene que recorrer los 1300 millones de tweets y crear una única tabla de hash. Recuerde que puede usar diagramas, esquemas y pseudocódigo para hacer más clara su explicación. Debe ser claro indicando cómo queda conformado el esquema LSH que va a usar en el punto 3.(35 puntos)

3. Describir cómo se hace la predicción de si un tweet es fake news o no utilizando el esquema LSH propuesto. (35 puntos)

## Datos del enunciado

Semejanza >= 0.73 -> Fake news.

Semejanza >= 0.55 -> 70% de probabilidad de colisión.

Semejanza <= 0.05 -> Menos de 1% de probabilidad de colisión

1. Teniendo los datos del enunciado se puede partir a calcular b y r en función de d1, d2, p1, y p2. Entonces:

   Mirando los datos enunciados en este examen, se ve que:
   - d1 = 1-0.55 = 0.45, que es la distancia para que puedan ser considerados como semejantes
   - d2 = 1-0.05 = 0.95, distancia para que se consideren como muy distintos.
   - P1 = 0.7, probabilidad de un 70% de colisión para semejantes.
   - P2 = 0.01, probabilidad de un 1% de que colisionen elementos muy disímiles.

   Entonces, la familia de hash nos queda definida como:

   H(0.45, 0.95, 0.7, 0.01)

   Para encontrar r y b entonces debería realizar grid search; por lo que realizo un código en Python(Aproximado debido a la falta de tiempo por el examen), y permitiéndome un cierto margen de error determinado en el mismo obtengo con el siguiente código:

```
d1=0.45
d2=0.95
p1=0.7
p2=0.01

r=1
b=1

p1temp = (1-(1-(1-d1)**r)**b)
p2temp = (1-(1-(1-d2)**r)**b)

while( (p1temp > (p1+0.04)) | (p1temp < (p1-
0.04))  | (p2temp > (p2+0.005)) | (p2temp < (p2-0.005))):
    if((p2temp > p2) & (p1temp < p1)):
        r+=1
        p1temp = (1-(1-(1-d1)**r)**b)
        p2temp = (1-(1-(1-d2)**r)**b)
    if (p1temp < p1):
```

```
        b+=1
    p1temp = (1-(1-(1-d1)**r)**b)
    p2temp = (1-(1-(1-d2)**r)**b)
    print ("P1: " + str(p1temp))
    print ("P2: " + str(p2temp))

print("r = " + str(r))
print("b = " + str(b))
print("p1 = " + str(p1temp))
print("p2 = " + str(p2temp))
```

El siguiente resultado:

r = 2

b = 3

p1 = 0.6606618906249999

p2 = 0.0074812656250001774

Es decir, con r=2 y b=3 puedo lograr aproximadamente el objetivo, con un margen de error de un 4% de probabilidad de desacierto en elementos similares, y un 0,3% de margen de error de probabilidad de que colisionen dos elementos muy disímiles.

Esta resolución aplica para un esquema AND de r (2 en este ejemplo) minhashes, con un OR de b (3 en este caso) grupos, bandas, o familias de hashing.

Entonces, para los falsos negativos podría decirse que como hay un 66% de probabilidad aproximada en mi ejemplo de que colisionen elementos con una similaridad de 0,55 según distancia Jaccard, entonces hay aproximadamente un 33% de probabilidades de que un elemento que este en nuestro rango de similaridad no colisione con ningún elemento con el cual si presenta similaridad.

Para los falsos positivos sería similar, pero en este caso, existiría una probabilidad no nula de que a una distancia mayor a 0,45(d1), se provoque una colisión, y podremos decir que entonces el elemento es similar a un fake news cuando en realidad no lo es. Sí en particular lo tomamos para muy disímiles, entonces un falso positivo se da cuando se cumple esta probabilidad p2.

Aplicando los porcentajes a los que llegue, de 10000 tweets:

- Falsos positivos:  0,7% * 10000 = 70 tweets serían posibles falsos positivos.
- Falsos negativos: 33% * 10000 = 3333 tweets serían posibles falsos negativos.

2. Para la etapa de preprocesamiento se debe tomar cada uno de estos 1300 millones de tweets, y hashearlos a partir de funciones de minhash, para lo cual se utilizarían n-shingles debido a que utilizaré distancia Jaccard. Estos n-shingles deben ser tomados de una longitud adecuada para que sean representativos; por ejemplo, shingles de 10 caracteres de longitud. De esta forma, se crea una matriz del estilo:

| | D1 | … | Dn |
|---|---|---|---|
| S1 | 1 | … | 0 |
| … | … | … | … |
| Sn | 0 | … | 1 |

En donde cada 1 representa que el shingle se encuentra presente en un documento, que en este caso sería un tweet cada uno de estos documentos.

Entonces, la cadena de texto correspondiente a un tweet, se descompondrá y obtendremos varios n-shingles de longitud en este caso 10, con los cuales armaremos la matriz dicha.

Una vez generada la matriz, tomo como funciones de minhash permutaciones aleatorias de esta matris, en donde los shingles se ordenen aleatoriamente. De esta forma, por ejemplo, un valor de minhash puede ser en que fila aparece el primer 1 para ese documento; y asi aplicarlo para r*b minhashes que me representarían la cantidad total de minhashes que necesito. Estos minhashes se generarían por ejemplo tomando el minimo hashing de un documento generado a partir de una permutación de filas.

Poseyendo los r*b minhashes generados a partir de permutaciones aleatorias de la matriz, puedo utilizar b familias de hashing, con las cuales unir estos minhashes. Estas serán del estilo:

$$\left(\left(\sum A.MHn + B\right) mod\ p\right) mod\ m$$

Donde A y B son dos constantes predefinidas, p es un numero primo mayor o igual, pero cercano a m, y m es la cantidad de buckets que utilizaremos para nuestro LSH.

De esta forma, cada familia de hashes, representaría una tabla en la cual se almacena el documento hasheado. Si un documento colisiona con una posición en la cual este elemento esté hasheado, puede ser considerado como similar. Para ser más estrictos o más laxos en la selección de similares, se puede operar entre conjuntos de varias familias de hashes con los operadores de conjuntos OR o AND. Utilizar AND reduce la cantidad de falsos positivos, pero dejando fuera algunos casos que si serían positivos, y OR nos flexibiliza mas el método reduciendo los falsos negativos, pero aumentando la cantidad de falsos positivos con los cuales podemos encontrarnos.

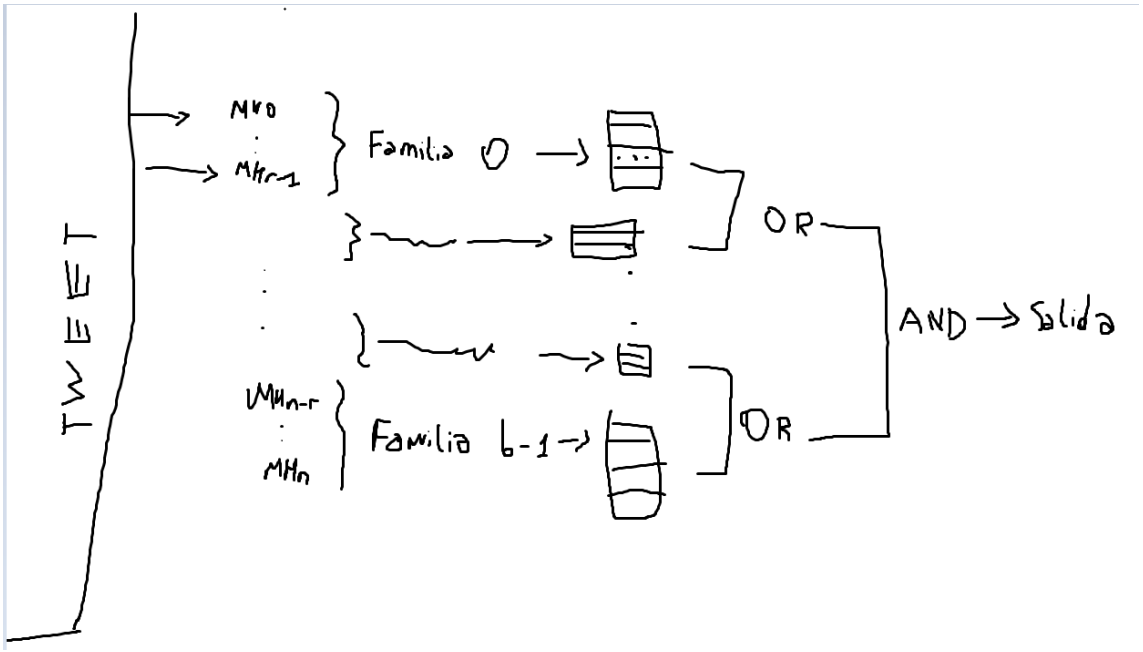
Entonces, finalmente, habiendo pasado cada uno de los tweets por los b*r minhashes, y aplicando las b familias de r minhashes cada una, podemos obtener las b tablas con las cuales podremos comparar un elemento query para ver si se asemeja o no.

Si buscamos realizar una única tabla, entonces bastaría con almacenar cada documento en cada una de las posiciones que asignan las b familias de hashing. Por ejemplo: Si la familia 0 otorga la posición 0, y la familia 1 otorga la posición 4, entonces el tweet se encuentra en la posición 0, y en la posición 4.

3. Dado el sistema de LSH propuesto en el punto 2, analizar la similaridad de un elemento query (Tweet que busca verse si es semejante o no a un fake news) es simple.

   Debe tomarse este elemento de la misma forma que se tomo cualquiera de los 1300 millones de tweets en el preprocesamiento, y se lo minhashea de la misma forma. Habiendo minhasheado el mismo, se recurre a las b familias de hashing, y se analiza la colisión que esta otorga en la tabla correspondiente a esa familia de hashing. En caso de colisión, se dice que a través de esa familia de hashing, el tweet es similar a alguno de los 1300 millones almacenados. En este caso, sería similar a aquellos que se encuentren en el bucket al cual se hasheo el tweet en dicha familia.

   Para evitar falsos positivos y falsos negativos, recurro a realizar la operación de que de esas b familias, puedo operar con AND entre algunas de las mismas, y efectuar ORS entre otras. Algo del estilo:




La salida poseería todos aquellos tweets a los cuales el tweet query es similar. En caso de ser un conjunto vacío, el tweet query no sería similar a ningún otro tweet, mientras que en caso de ser un conjunto no vacío, el tweet query presentaría similaridad con los elementos

presentes en el conjunto, y debemos analizar que % de similaridad posee con los mismos. De esta manera, evitamos comparar con los 1300 millones de tweets presentes, y comparamos únicamente con los similares.

Si se cumple que el % de similaridad es mayor o igual a 1-d1 enunciada, entonces el tweet sería tomado como un fake news. Si no se cumpliese que con alguno de los elementos del conjunto la similaridad es mayor o igual a 1-d1, entonces el tweet no sería fake news.

Utilizando una tabla única el comportamiento es similar solo que en vez de operar entre b tablas, se opera con las b posiciones que nos otorguen las familias de hashing. Es decir, si obtenemos los tweets correspondientes a las posiciones 1, 5, y 20, podemos operar con los operadores OR y AND para obtener el conjunto final de elementos con los cuales comparar 1 a 1 nuestro elemento query.