Clasificación de textos
Usando Matriz-Vector Sparse Multiplicación
David Buitrago Arenas
Trabajo para asignatura LAPPAD – Profesor Vicente Vidal

Resumen
Se presenta el problema de indexar grandes cantidades de documentos en un tiempo corto de una forma eficaz, la información que contienen es importante y los métodos de los que se dispone actualmente favorecen los objetivos de clasificación por palabra clave, por nombre, por dato especifico, donde los resultados para estos son bueno pero presentan una perdida de información valiosa en el camino hacia la clasificación.
El objetivo principal del experimento es entender la eficacia de la multiplicación de matrices de n x m dimensiones, que nos dan para generar clasificaciones, muy veraces.
En este informe se detalla un ejemplo de ejercicio de clasificación y luego se hace una extensión sobre los experimentos que se desarrollaron con el sistema que se implemento y el corpus que se utilizo.

Introducción

Las complicaciones para indexar y tratar colecciones de datos, han tenido en principio como solución la implementación de métodos como QR [1] y como se muestra en este informe la implementación del SVD(Singular Value Descomposition), para hacer un LSI(Latent Semantic Indexing).
El más reciente método de indexación semántica latente (LSI) o análisis semántico latente (LSA) es una variante del modelo de espacio vectorial en el que se emplea una aproximación de bajo rango a la representación espacio vectorial de la base de datos [2,3]. En este caso incluyo lsi_classificator.py que esta basado en [1] que es el experimento básico, por otra parte incluyo la variante con la introducción de 4 documentos con 1000 textos de tweets relacionados con sentimientos, etiquetados por #Amo, #Feliz, #Triste, #Miedo y una pregunta basica que en la experimentación he ido variando para comprobar la eficacia de la clasificación basada en la multiplicación de matrices.

Experimentación

- Los stop-words se mantienen
- El texto es tokenizado y pasado a minúsculas
- No se hace uso de steaming
- Los términos son organizados alfabéticamente

Explicación de la implementación
Ejemplo: Se tiene una colección con los documentos descritos por las siguientes frases.

d1: *Shipment of gold damaged in a fire.*
d2: *Delivery of silver arrived in a silver truck.*
d3: *Shipment of gold arrived in a truck.*

Y la pregunta es q="gold silver truck"

Paso 1. Hacemos un termino de frecuencia por documento y incorporamos la información en nuestra matriz, documentos vrs palabras, de la misma forma hacemos sobre el vector de q.

Terms		d1	d2	d3		q
a		1	1	1		0
arrived		0	1	1		0
damaged		1	0	0		0
delivery		0	1	0		0
fire		1	0	0		0
gold	**A =**	1	0	1	**q =**	1
in		1	1	1		0
of		1	1	1		0
shipment		1	0	1		0
silver		0	2	0		1
truck		0	1	1		1

Paso 2. Descomponer matriz A y buscar el U, S, V de la implementación del método SVD.

$$A = USV^T$$

Donde tendremos unas matrices con una información como:

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

Paso 3. Se implementa el ranking k=2, donde reducimos la dimensión de las matrices,

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \qquad k = 2$$

$$S \approx S_k = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

$$V \approx V_k = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} \qquad V^T \approx V_k^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix}$$

Paso 4. Se hallan los vectores por documento

d1(-0.4945, 0.6492)
d2(-0.6458, -0.7194)
d3(-0.5817, 0.2469)

Paso 5. Encontrar el vector de q sobre el conjunto indexado.

$$q = q^T U_k S_k^{-1}$$

$$q = q^T U_k S_k^{-1} \qquad k = 2$$

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$q = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

Paso 6. Calculo de la similaridad

$$sim(q, d) = \frac{q \bullet d}{|q||d|}$$

$$sim(q, d_1) = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

$$sim(q, d_2) = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$sim(q, d_3) = \frac{(-0.2140)(-0.5817) + (-0.1821)(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$d_2 > d_3 > d_1$$

El ejercicio que se desarrollo es el siguiente:

Se presentan 4000 tweets, separados en 4 archivos donde están separados por el etiquetado de los hashtags de twitter *d1*=#Amo, *d2*=#Feliz,*d3*=#Miedo y *d4*=#Triste y se presenta la pregunta *q*=" #QuiebreTotal cuando me acuerdo ese día en la casa de la maca #no #no #noooooooooooooo #triste".
Se hacen los pasos en la implementación antes descritos.

Dimensiones de las matrices

q=(12037, 1)

U=(12037, 12037)

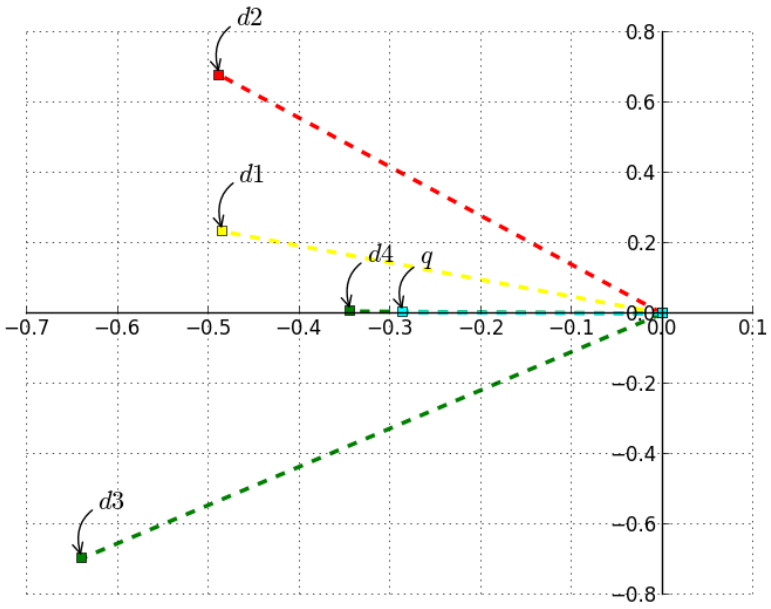
V=(4, 12037)

Los vectores resultantes por documento:
d1: [-0.48445098 0.23326451]
d2: [-0.4877878 0.67811607]
d3: [-0.63972055 -0.69692928]
d4: [-0.34369745 0.00598785]
El ventor q = [[-0.000888072234905961], [-0.00034276716814784 21]]

Resultados de la similaridad entre q y dn

Sim(q,d1): 0.684344503089
Sim(q,d2): 0.252466953321
Sim(q,d3): 0.896132026414
Sim(q,d4): 0.926508438139

Donde encontramos que d4 > d3> d1 > d2 , por tanto q es de la clase d4 y las palabras que expresa el vector q pertenecen a la clasificación de #Triste.



Experimentos adicionales

Inicial

q=" #QuiebreTotal cuando me acuerdo ese día en la casa de la maca #no #no #noooooooooooooo #triste".

d1: 0.718493681116
d2: 0.298536443229
d3: 0.873840369856
d4: 0.943472271099

Exp2.
Quitamos la palabra #triste
q=" #QuiebreTotal cuando me acuerdo ese día en la casa de la maca #no #no #noooooooooooooo".

d1: 0.684344503089
d2: 0.252466953321
d3: 0.896132026414
d4: 0.926508438139

Exp3.
Quitamos la palabra #no #no #noooooooooooooo
q=" #QuiebreTotal cuando me acuerdo ese día en la casa de la maca"

d1: 0.675872760282
d2: 0.24126859383
d3: 0.901200532117
d4: 0.922098409303

Sobre los experimentos iniciales generamos un listado de 20 tweets por documentos seleccionados randomicamente y los pasamos por el sistema para clasificar y medir la precisión.

El tiempo de clasificación por documento es similar, entonces algunas de las modificaciones importantes fue la de guardar la matriz A en un archivo lo mismo para las matrices derivadas de la implementación del SVD.
Esto redujo los tiempos en un 30% total.

En los resultados del experimento con el listado de los 20 tweets por emoción, se logro un 92% de precisión sobre el tweet sin modificar al clasificarlo.

Se encontró que a pesar de indexar palabras por hash tag en español algunos tweets estaban en portugués, en francés y hasta en ingles, donde incorporaban el hash tag indicado en castellano.

Conclusiones y trabajo futuro

Fue muy interesante, entender el procedimiento de la implementación de el algoritmo de SVD y lograr el LSA.
En principio se realizaron ejercicios en papel y lápiz, luego se traslado el mismo resultado a el ordenador para contrastarlos con un numero de datos importantes y un tamaño de matriz considerable.

En trabajos futuros, se podría realizar muchas mas experimentaciones en principio realizar la experimentación para un set de ejemplos igual al del ultimo experimento, incorporando la excepción o borrado de las palabras hash del tweet para ver la capacidad de clasificación con bolsa de palabras.

Algo interesante también podría ser pasar algún algoritmo de hashing antes de la implementación del SVD , después de la implementación de un tfidf y ver como van los resultados de LSA.

Por otra parte hacer los experimentos necesarios para incorporar dimensiones a las matrices desde un análisis POS por frase , por documento. Para ver si esto puede mejorar de alguna forma la clasificación y presentar una ayuda en temas como análisis basado en una estructura sintáctica particular basado en valores morfosintácticos.

Referencias

[1]Matrices, Vector Spaces, and Information Retrieval, Michael W. Berry 1999
[2] M. W. Berry, S. T. Dumais, and G. W. O'Brien, Using linear algebra for intelligent infor- mation retrieval, SIAM Rev., 37 (1995), pp. 573–595.
[3] M. Berry and R. Fierro, Low-rank orthogonal decompositions for information retrieval ap- plications, Numer. Linear Algebra Appl., 3 (1996), pp. 301–328.