

UNIVERSIDAD DIEGO PORTALES

Tarea 4: Minería de Datos y Procesamiento de Lenguaje Natural

Rodrigo Fuenzalida

Profesor: Alejandro Figueroa

Ayudante: Nicolás Olivares

Facultad de Ingeniería

Escuela de Informática y telecomunicaciones

30 de septiembre de 2013

Índice

1. Introducción	1
2. Desarrollo	2
2.1. Parte 1	2
2.1.1. Resultados	3
2.1.1.1. Pregunta 1	3
2.1.1.2. Pregunta 2	4
2.2. Parte 2	5
2.2.1. Resultados	5

Capítulo 1

Introducción

El objetivo de esta tarea es analizar y comprender el uso y funcionamiento del algoritmo de clustering DBSCAN para así darle un uso útil en esta tarea.

Este trabajo cuenta con 2 partes, la primera de estas es analizar el comportamientos del algoritmo DBSCAN y la segunda en el etiquetado sintáctico de las consultas.

Capítulo 2

Desarrollo

2.1. Parte 1

En la primera parte de esta tarea extenderemos nuestro conocimiento de los algoritmos de clustering vistos en la tarea anterior (i.e., K-Means y Fuzzy C-Means) a DBSCAN. Una de las diferencias de este método con los dos anteriores es que no necesita el parámetro k para ejecutarse. No obstante, necesita otros dos parámetros: minPts que es el número mínimo de puntos y ϵ que es el radio de la vecindad. Ambos parámetros tienen como objetivo definir qué es interpretado como denso o parte de un clúster. Para esta tarea, el alumno puede utilizar cualquier implementación de DBSCAN que consiga en Internet.

Utilizando los vectores compuestos por los lexemas entregados por MontyLingua, se pide:

1. Ejecutar DBSCAN con diferentes parámetros minPts (entre 1 y 10) y ϵ (entre 1 y 3). Comentar y mostrar los resultados en una tabla. Utilizar como métrica Accuracy.
2. Los puntos asignados como "NOISE" deben asumirse como incorrectamente clusterizados. Comente y muestre este último punto en su reporte.
3. Probar con diferentes métricas de distancia. Para esto sólo utilice las implementadas por la librería que opte por utilizar.

Nótese que es esperable que los resultados no sean satisfactorios y tiendan a ser confusos. Intente explicar las razones de esto, especialmente analizando lo que DBSCAN intenta hacer. Nótese también que en caso de no poder lematizar las palabras, se puede utilizar el vector no-lematizado (palabras tal cual vienen en las consultas), pero eso se considerará en la evaluación de la tarea.

2.1.1. Resultados

2.1.1.1. Pregunta 1

En este caso se utilizó una implementación de la biblioteca "sklearn" la cual pertenece al lenguaje de programación Python y que posee una implementación del algoritmo DBSCAN. A continuación se muestran los resultados obtenidos con las consultas lematizadas y los distintos valores de ϵ y minPts:

Eps	MinPts	Accuracy
1	1	0
1	2	0.5
1	3	0.5
1	4	0.5
1	5	0.5
1	6	0.5
1	7	0.5
1	8	0.51
1	9	0.51
1	10	0.52
2	1	0.31
2	2	0.32
2	3	0.32
2	4	0.32
2	5	0.32

FIGURA 2.1: 15 Primeros resultados

Eps	MinPts	Accuracy
2	6	0.32
2	7	0.32
2	8	0.32
2	9	0.32
2	10	0.32
3	1	0.31
3	2	0.31
3	3	0.31
3	4	0.31
3	5	0.31
3	6	0.31
3	7	0.31
3	8	0.31
3	9	0.31
3	10	0.31

FIGURA 2.2: Últimos 15 resultados

Se aprecia que a valores más bajos de ϵ y minPts se obtienen valores de accuracy mucho más altos ya que los valores obtenidos son más exactos que en valores muy altos.

2.1.1.2. Pregunta 2

Dado los parámetros de entrada que necesita DBSCAN (minpts y ϵ), se puede apreciar que el ruido o noise(datos errados), varía de acuerdo a estos parámetros los cuales le dan una mayor o menor sensibilidad a los puntos obtenidos por el algoritmo. Se puede apreciar que a valores más alto de ϵ y del minPts se hace más estable DBSCAN, entregando mucho menos NOISE.

Eps	MinPts	NOISE
1	1	1720
1	2	8
1	3	0
1	4	0
1	5	0
1	6	0
1	7	0
1	8	0
1	9	0
1	10	0
2	1	33
2	2	0
2	3	0
2	4	0
2	5	0

FIGURA 2.3: 15 Primeros resultados de NOISE

Eps	MinPts	NOISE
2	6	0
2	7	0
2	8	0
2	9	0
2	10	0
3	1	0
3	2	0
3	3	0
3	4	0
3	5	0
3	6	0
3	7	0
3	8	0
3	9	0
3	10	0

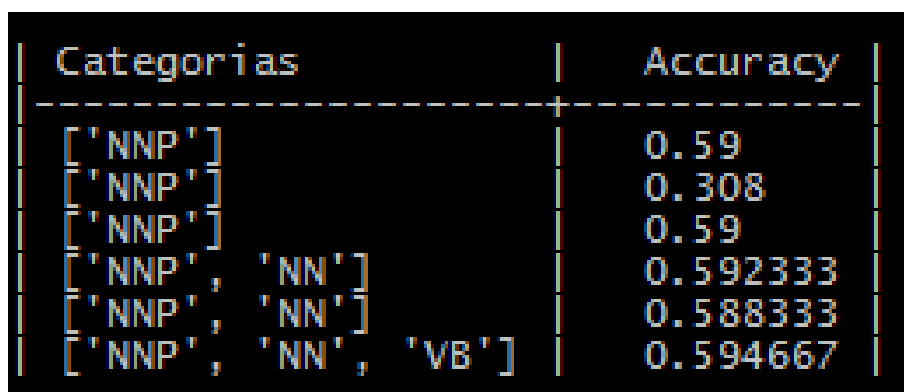
FIGURA 2.4: Últimos 15 resultados de NOISE

Como se aprecia en las imagenes en la tercera columna se indica la cantidad de consultas que pertencen a la categoría NOiSE lo cual nos da un indicio de los resultados que se obtienen con los distintos parámetros para este método de clustering.

2.2. Parte 2

Con Stanford POS Tagger se obtuvieron las categorías de cada consulta, con lo cual se desarrollo el algoritmo para ver la predominancia de las categorías del vector original X:
 $X = [\text{NNP}, \text{NN}, \text{NNS}, \text{JJ}, \text{CD}, \text{VB}]$

2.2.1. Resultados



Categorias	Accuracy
['NNP']	0.59
['NNP']	0.308
['NNP']	0.59
['NNP', 'NN']	0.592333
['NNP', 'NN']	0.588333
['NNP', 'NN', 'VB']	0.594667

FIGURA 2.5: Resultados del algoritmo greedy

Como se puede apreciar en la tabla a medida que avanzan las iteraciones se van agregando más categorías a medida que el accuracy aumenta, esto aumenta porque se va agregando mayor información a las consultas, lo cual las hace más rica en información y proporciona mayor exactitud al momento de usar K-Means.