Trabajo Práctico: Fine Food Reviews
Organización de Datos 75.06

Joaquín Blanco, Padrón: 94653
Joaquín Casal, Padrón: 98280
Franco Etcheverri, Padrón: 95812
Agustín Luques, Padrón: 96803

Nombre del grupo en Kaggle: Tony Spark
Número del grupo: 19

21 de noviembre de 2016

Índice general

Resumen

En el presente documento se propone explicar la forma en la cuál se llevó adelante la resolución del trabajo práctico y todas las opciones evaluadas durante el desarrollo del mismo.

La base de las ideas que fueron presentadas en el informe de diseño son las mismas sobre las cuales se sustentan nuestros algoritmos finales. En las siguientes secciones se desarrollarán en mayor detalle los cambios y mejoras que se implementaron.

En términos generales, para la resolución del problema planteado se utilizan dos algoritmos principales:

- Algoritmo de estilo probabilístico (Adaptación de Naive Bayes)
- Algoritmo de K-Means

Ambos algoritmos son independientes uno del otro y generan su propia predicción sobre la review. Por este motivo y con el objetivo de llegar a una precisión mayor se desarrollaron una serie de combiners los cuales a partir de los dos resultados generasen uno nuevo como valor final.

Capítulo 1

Análisis inicial de los datos

Como bien se planteo desde un principio el campo principal sobre el cual íbamos a enfocar nuestro desarrollo era el de texto. Las principales diferencias con respecto al diseño son que, luego de estudiar nuevamente el set de datos, consideramos que los "Helpfullnes Numbers"no iban a servir para la tarea que teníamos que llevar adelante. Dichos campos serán más apropiados para un caso en el cual hubiésemos tenido que otorgarle un puntaje al producto y no a la review. En dicho caso, estos valores hubieran jugado un rol importante, pero para nuestro análisis fueron descartados.

En resumen, los únicos campos que tomamos en cuenta, obviamente además de la predicción del set de training, fueron el text y el summary.

Capítulo 2

Transformando los datos

Como bien se presento en el diseño, la idea era pre-procesar los textos para quedarnos únicamente con aquellas palabras que mejoraran el aprendizaje del algoritmo y aseguraran un correcto funcionamiento.

Si bien esto se hizo para ambos algoritmos, para el algoritmo de Bayes se hizo una diferenciación. La diferencia frente a lo planteado inicialmente es que las únicas palabras que afectarán la predicción final de dicho algoritmo son los adjetivos, por lo que al resto no se les va a calcular un puntaje asociado y a la hora de hacer predicciones, las palabras que no sean adjetivos son ignoradas. Para ello se utilizó una lista de adjetivos en inglés.

Capítulo 3

Algoritmo de estilo probabilístico (Adaptación de Naive Bayes)

La clasificación de Bayes consiste en asignarle la clase con mayor probabilidad a un ejemplo determinado a partir de su contenido. Para adaptar el algoritmo a nuestro trabajo lo que se busco es extender la posibilidad de clasificación más allá de las clases sobre las que partimos, en nuestro caso la cantidad de estrellas del review del 1 al 5. Para ello, y con el objetivo de aceptar valores no enteros como resultado, se propuso obtener la predicción a través de un promedio de los puntajes asociados a cada palabra, más precisamente adjetivo, de la review.

Los pasos del algoritmo son los siguientes:

1. Se crea un diccionario inicialmente vacío y una lista con las palabras de negación en ingles (Ej: not, isnt).
2. Se abre un archivo de texto en el cual se encuentran guardados todos los adjetivos que vamos a considerar.
3. Se va guardando en el diccionario los adjetivos y adicionalmente una combinación de negación+adjetivo (concatenados) para cada adjetivo y cada palabra de la lista de negaciones.
4. Abrir el archivo de train (previamente pre procesado) y por cada fila únicamente toma los campos de texto y resumen.
5. Tomar cada palabra de los campos anteriores y si ésta se encuentra en el diccionario creado, se la agrega a una nueva lista de la siguiente forma: (Puntaje del review actual, palabra).
6. Se genera un RDD con la lista completa de palabras con sus respectivos puntajes.
7. Map para darle el siguiente formato a los datos: (Palabra, (1, puntaje)).

8. ReduceByKey para obtener la cantidad de veces que aparece cada palabra junto con la suma de puntajes de los reviews en las cuales aparece.

9. A través de otro Map se obtiene el puntaje promedio de cada palabra: (Palabra, Puntaje Promedio).

10. Abre el archivo de Test (previamente pre procesado) y coloca en un RDD (ID Review, texto)

11. Separa cada palabra del campo texto para lograr el siguiente formato: (Palabra, ID)

12. A través de un leftOuterJoin entre los dos RDDs se logra generar la tupla (Palabra, (ID, Puntaje)) en el caso de que la palabra se encuentre en el RDD de train y (Palabra, (ID, none)) en caso contrario.

13. Se elimina el campo de palabra del RDD.

14. Map para lograr (ID, (Puntaje, 1)).

15. ReduceByKey para obtener la cantidad de palabras consideradas para predecir el puntaje junto con el puntaje total obtenido de la suma del de cada palabra.

16. Map para lograr el formato de predicción correspondiente: (ID, Puntaje Promedio).

Capítulo 4

Hashing Trick, K-means y Knn

Partiendo de los datos presentes ya procesados, se decidió usar el método de Hashing Trick para modelar los datos en un formato vectorial y así poder usarlos con Kmeans. Respecto de la dimensionalidad a la cual proyectar cada uno los textos, mediante pruebas se obtuvo aquella que sea tratable en memoria dado los recursos tecnológicos con los que se cuenta y que genere buenos resultados en el algoritmo que se utilizara luego.

Mediante un análisis con Kmeans++ observamos que con pocas dimensiones, los datos tendían a acumularse en pocos cluster. Esto fue un gran problema dado que nuestra propuesta era la de usar Kmeans para mitigar el costo de ejecutar KNN. Si todos los datos confluían en pocos grupos, Kmeans no surtiría el efecto esperado.

Incrementando paulatinamente la cantidad de dimensiones llegamos a las que se encuentran presentes en las pruebas sobre kaggle (101 y 53 dimensiones, siendo 101 la cantidad de dimensiones mas usada).

Respecto de la selección de centroides iniciales, se había planteado en el diseño inicial hacer uso del método presente Kmeans++. Sin embargo, debido al costo de calcular algunos pocos centroides y los resultados poco confiables se optó por producir centroides tomando, de forma aleatoria, vectores del set de train. Con esto se logró una mejor distribución de los datos en los distintos clusters, reduciendo el tiempo de ejecución de Kmeans.

Por lo dicho anteriormente, decimos entonces que, para determinar el puntaje de una review, primero se la pre-procesa con métodos ya mencionados en este informe. Luego se le aplica Hashing trick para obtener un vector representativo del texto presente en el registro. Mediante kmeans ya entrenado, se determina el cluster mas cercano y se compara nuestro vector con todos los vectores que se encuentran en el cluster para llegar a los k vecinos mas cercanos.

Finalmente, mediante un promedio de los puntajes de esos k vecinos mas cercanos, se llega al puntaje a asignar a la review inicial.

Capítulo 5

Combinación de clasificadores

Para lograr combinar los resultados de cada algoritmo independiente, se utilizaron los siguientes métodos de combiners:

- Promedio
- Promedio ponderado

Para el caso del promedio ponderado, se utilizó un factor de fiabilidad para darle mayor importancia a la predicción de uno u otro algoritmo.

La idea original para obtener el valor de fiabilidad era separar una porción del set de entrenamiento y dejarla fuera del entrenamiento de los algoritmos para luego poder evaluar que tan preciso son los mismos. Esto es posible ya que la idea era predecir los puntajes de las reviews de la porción del set de entrenamiento excluída, de las cuales sabemos la verdadera puntuación.

Sin embargo, teniendo en cuenta los tiempos de ejecución que implica nuestro algoritmo de Kmeans, nos volcamos por otra alternativa: calcular los pesos según el puntaje de cada algoritmo en Kaggle con la siguiente fórmula:

$$Peso\ algoritmo = 1 - (Puntaje\ del\ algoritmo\ en\ kaggle\ /\ Puntaje\ total\ sumando\ los\ dos\ algoritmos)$$

A partir de los resultados obtenidos y normalizando los valores para que la suma de los dos factores de fiabilidad (uno para cada algoritmo) de igual a uno, se procedió a calcular la predicción final como:

$$Predicción = C_1.Predicción_1 + C_2.Predicción_2$$

Capítulo 6

Tabla de Submits de Kaggle

Submit	Descripción	Resultado
1	Bayes: sin juntar negaciones	1.48598
2	Bayes: juntando negaciones	1.46913
3	Bayes: agregamos campo summary, valor por default 4.5	1.38063
4	Bayes: agregamos campo summary, valor por default 4.0	1.37189
5	Bayes: agregamos campo summary, valor por default 3.0	1.37214
6	Kmeans: dim: 101, clusters: 1000, knn: 7	1.87268
7	Kmeans: dim: 53, clusters: 2000, knn: 7	2.19731
8	Kmeans: dim: 101, clusters: 2000, knn: 7	1.48906
9	Kmeans: dim: 101, clusters: 3000, knn: 7	1.48631
10	Combiner: combinación del mejor bayes y mejor kmeans	1.29659

Cuadro 6.1: Tabla simplificada.

Capítulo 7

Comentarios

La idea de tener que predecir en función de lo que un usuario escribe nos resultó muy interesante pero a la vez un desafío algo difícil, más que nada por el hecho de que las personas muchas veces realizan errores de tipeo o tienen faltas de ortografíŋa que si bien son pequeños detalles, en cuanto a la forma de detectarlos y evitar que jueguen un papel negativo importante en el desarrollo de nuestros algoritmos es algo vital y bastante complicado de solucionar por completo.

Por otro lado, nos resultó sorprendente la inmensa variedad de enfoques que se le pueden dar a temas relacionados con text mining. Ya que más allá de los algoritmos que elegimos nosotros, en la parte previa de estudio y diseño del TP fuimos discutiendo y evaluando muchos más.

Los resultados creemos que son positivos, evaluando el tiempo acotado que se tiene en solo un cuatrimestre, la falta de recursos más que nada por el lado de la cantidad de tiempo que llevaba correr el Kmeans en nuestras computadoras y la poca experiencia de nuestra parte para con estos temas en un principio, el saldo final es bueno.

En conclusión fue un trabajo interesante y llevadero, el cual si bien es complicado y lleva su tiempo, conlleva consigo mucho aprendizaje y satisfacción a la hora de ver los resultados obtenidos.

Bibliografía

[1] Bing Liu, Lei Zhang. A Survey of Opinion Mining and Sentiment Analysis, Capitulo del libro Mining Text Data. Ed. C. Aggarwal, C. Zhai, Springer, 2011.

[2] Leah S. Larkey, W. Bruce Croft. Combining Classifiers in text categorization. Departamento de Ciencias de la Computación. Universidad de Massachusetts.

[3] Daphne Koller, Mehran Sahami. Hierarchically classifying documents with very few words. Departamento de Ciencias de la Computación. Universidad de Stanford

[4] Editors: Aggarwal, Charu C., Zhai, ChengXiang (Eds.). Mining Text Data

[5] Paul N. Bennett, Susan T. Dumais, Eric Horvitz. Probabilistic Combination of Text Classifiers Using Reliability Indicators: Models and Results.

[6] Ronen Feldman, James Sanger. The Text Mining Handbook.

[7] Kunpeng Zhang, Yu Cheng, Wei-keng Liao, Alok Choudhary. Mining Millions of Reviews: A Technique to Rank Products Based on Importance of Reviews. Universidad de Northwestern.

[8] Luis Argerich, Apuntes del Curso Organización de Datos.

[9] Data Science in Minutes. `https://rdisorder.wordpress.com/2016/08/06/data-science-in-minutes/`

[10] All About Stop Words for Text Mining and Information Retrieval. `http://text-analytics101.rxnlp.com/2014/10/all-about-stop-words-for-text-mining.html`

[11] Clasificador bayesiano ingenuo. `https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo`

[12] Listado de adjetivos en inglés `http://www.enchantedlearning.com/wordlist/adjectives.shtml`