U.B.A. FACULTAD DE INGENIERÍA

75.06 - Organización de Datos

"*When Bag of Words meets bags of Popcorn*"

1er cuatrimestre de 2015

Docente a cargo: Luis Argerich
Nombre del grupo: Random Typing Monkeys

Acuña, Santiago	95313	acunia.santiago@gmail.com
Ciruzzi, Pablo	95748	rubic9@hotmail.com
Landino, Francisco	94475	landinofrancisco@gmail.com
Rupcic, Florencia	94525	florencia441@hotmail.com

Nombre del grupo en Kaggle: Random Typing Monkeys

Descripción Breve:

Para la resolución del trabajo se decidió utilizar distintos algoritmos con el objetivo de determinar si una review es positiva o negativa. Cada uno de los algoritmos devolverá la probabilidad de que una review sea positiva. Realizando una suma ponderada de las probabilidades devueltas por todos los algoritmos, se determinará finalmente si la review es positiva o negativa.
Para realizar dicha ponderación se pueden comparar los resultados de las predicciones usando cross validation, y luego, teniendo una idea de qué pesos iniciales asignarle a cada algoritmo, se perfeccionaran los pesos usando un algoritmo para adaptarlos.

Fuentes/Bibliografía:

Bag of Words model:

- http://en.wikipedia.org/wiki/Bag-of-words_model

Naïve Bayes Classifier:

- "Naïve Bayes and Text Classification", Sebastian Raschka: http://sebastianraschka.com/Articles/2014_naive_bayes_1.html
- http://en.wikipedia.org/wiki/Naive_Bayes_classifier

Perceptrón:

- "Online Learning Perceptron", MLWave: http://mlwave.com/online-learning-perceptron/

Explicación de la solución:

Algoritmo 1: "Más y menos uno"

Paso 1: Creación del Bag of Words

Se crea un diccionario y dos vectores, uno para las reviews positivas y otro para las negativas. Las claves del diccionario son las palabras que aparecen en las reviews de entrenamiento, y los valores son índices. El índice marca en qué posición de ambos vectores se encuentra la frecuencia con la que aparece cada palabra del Bag, en las reviews positivas y en las reviews negativas de entrenamiento.
Este paso se encuentra mejor desarrollado en la sección donde se explica cómo se procesan las reviews.

Paso 2: Predicción de reviews

Se busca cada palabra de la review a predecir dentro del Bag y se obtiene el índice del vector de frecuencias. Se accede al vector en dicha posición y se obtiene con qué frecuencia aparece la palabra, en las reviews negativas y en las reviews positivas. Si la palabra aparece con mayor frecuencia en las reviews positivas, se suma 1 a un contador, y si aparece con más frecuencia en las reviews negativas, se resta 1 al contador.

Al finalizar de analizar todas las palabras de la review, se observa el contador. Si el contador es mayor a cero, se devuelve con probabilidad igual a 1 que la review es positiva. Si el contador es menor a cero se devuelve con probabilidad igual a 0 que la review es positiva. Por último, si el contador es igual a cero, se devuelve con probabilidad igual a 0,5 que la review es positiva.

Algoritmo 2: "Palabras pesadas"

Paso 1: Creación del Bag of Words

Este paso es igual al del Algoritmo 1, solo que luego de terminar de llenar el Bag y los dos vectores de frecuencias, se le asigna un peso a cada palabra.
Se construyen otros dos vectores, donde en vez de aparecer la frecuencia con la que aparece cada palabra, se va a almacenar el peso de cada palabra. De esta forma, a las palabras que tengan mayor frecuencia, en cada uno de los vectores se les va a asignar un valor más grande (Ej: 10) y a las palabras que tengan menor frecuencia en cada uno de los vectores se les va a asignar un valor más chico (Ej: 1).

Paso 2: Predicción de reviews

Al igual que en la Algoritmo 1, se analiza palabra por palabra de la review a determinar, obteniendo el índice del Bag y luego accediendo al vector que contiene los pesos de cada palabra. Se suman o restan dichos pesos al contador, dependiendo de si se obtienen del vector con pesos para las reviews positivas o para las reviews negativas respectivamente. Finalmente se devuelven las mismas probabilidades que en la Algoritmo 1, dependiendo de si el contador quedó mayor, menor o igual a cero.

Algoritmo 3: "Naïve Bayes"

Paso 1: Creación del Bag of Words

Es igual al Paso 1 del Algoritmo 1.

Paso 2: Predicción de reviews

Para cada review a analizar se calcula y se devuelve la probabilidad de que sea positiva usando un modelo probabilístico de Bayes.

X: la review es positiva
Y: la review es negativa
W_i : palabra *i* en la review a analizar, con *i* = 1...*n* y *n* = cantidad de palabras en la review

$$P(X \mid Words) =$$

$$= \frac{P(W1|X) * P(W2|X) * ... * P(Wn|X) * P(X)}{P(W1|X) * P(W2|X) * ... * P(Wn|X) * P(X) + P(W1|Y) * P(W2|Y) * ... * P(Wn|Y) * P(Y)}$$

Donde:

$$P(Wi \mid X) = \frac{frecuencia\ word_i\ en\ reviews\ positivas}{apariciones\ totales\ de\ word_i}$$

$$P(Wi \mid Y) = \frac{frecuencia\ word_i\ en\ reviews\ negativas}{apariciones\ totales\ de\ word_i}$$

Debido a que las probabilidades P(Y) y P(X) valen siempre 0,5, se podrían eliminar de la ecuación para evitar hacer más cálculos.
Un problema de esta fórmula es que se están separando las probabilidades P(Wi | X ó Y) con la justificación de que la probabilidad de que aparezca cualquier palabra es independiente de otra palabra, lo que se sabe que en realidad no se es asi. Según la bibliografía leída en la práctica Naive Bayes sigue mostrando buenos resultados a pesar de esto. De todas formas se podrían usar bigramas o buscar algun otro modelo que resuelva dicho problema.

Ejemplo:

Analizó la review “What a good movie”. Las palabras “what” y “a” son eliminadas porque son stopwords.
La palabra “good” aparece 700 veces en las reviews positivas y 200 veces en las reviews negativas.
La palabra “movie” aparece 500 veces en las reviews positivas y 600 veces en las reviews negativas.
Recordar que las probabilidades P(X) y P(Y) no son incorporadas a la ecuación porque ambas valen 0,5.
Las probabilidades quedarían:

P(“good” | X) = 700 / 900
P(“movie” | X) = 500 / 1100

P(“good” | Y) = 200 / 900
P(“movie” | Y) = 600 / 1100

$$P(X \mid Words) = \frac{P(\text{“good”} \mid X) * P(\text{“movie”} \mid X)}{P(\text{“good”} \mid X) * P(\text{“movie”} \mid X) + P(\text{“good”} \mid Y) * P(\text{“movie”} \mid Y)}$$

$$P(X \mid Words) = \frac{\frac{700}{900} * \frac{500}{1100}}{\frac{700}{900} * \frac{500}{1100} + \frac{200}{900} * \frac{600}{1100}} = \frac{35}{47} = 0,7447$$

La review es calificada por este algoritmo como positiva.

Algoritmo 4: “Perceptrón”

Con el perceptrón se busca encontrar un valor numérico para una review, resultado del producto interno entre las palabras de esa review y sus respectivos pesos.
El producto interno se realiza, para la palabra, entre el resultado de aplicarle una función de hash y su peso correspondiente. En un principio los pesos se inicializan de manera aleatoria, o en algún valor fijado, como por ejemplo 0.
Durante la etapa de entrenamiento, se busca predecir el sentiment de acuerdo al valor dado por el producto interno, y se lo compara con el ya conocido. Si la predicción es errada, se recalculan los pesos, teniendo en cuenta dicha predicción. Si, en cambio, la predicción es correcta, los pesos se mantienen. Este proceso se da un número de veces fijado previamente.
Una vez terminado este proceso, se tienen los pesos que se consideran adecuados para hacer predicciones certeras, y se procede a predecir el resto de las reviews.

Resultado final: Ponderación

Para calcular la ponderación que va a tener cada algoritmo en la decisión de la probabilidad final sobre si una review va a ser positiva o negativa, se va a usar cross validation y también se va a calcular los pesos que va a tener cada algoritmo usando una adaptación de los pesos de cada probabilidad devuelta por los algoritmos.
Cross validation consiste en tomar diferentes particiones de reviews del labeledTrainData.tsv (archivo en el cual ya se conoce el sentimiento para cada review). Una porción del archivo se utilizará para entrenar al algoritmo. Sobre la otra porción del archivo, se predecirá el sentimiento de cada review. De esta forma se evita hacer un overfit del set de entrenamiento, ya que puede ser que un algoritmo funcione mejor que otro para una cierta partición de pruebas, pero peor que otro para otra partición de pruebas.
Luego de tener una cierta ponderación inicial sobre los algoritmos, se va buscar adaptar los pesos de cada algoritmo, usando la porción reservada para las pruebas, donde antes de aprender de cada review, se va a predecir su sentimiento con cada algoritmo. Dependiendo de si la predicción resultó o no exitosa, se va a modificar el peso de los mismos.

Explicación del proceso de los textos:

Para el procesamiento de las reviews se comenzará eliminando de las mismas todos los caracteres que no sean alfabéticos. Esto incluye números, signos de puntuación, caracteres especiales, y todo aquello que consideremos que pueda afectar nuestros resultados. Como caso particular, también se eliminarán las URLs y los tags HTML con las que cuentan las reviews a tratar.
Toda ‘stop word’, palabra de uso común que es considerada poco informativa (como por ejemplo artículos, pronombres y conjunciones), será borrada de las reviews.
Las palabras restantes serán pasadas a minúscula.

Para los primeros tres algoritmos se procederá de la misma manera.
Una vez parseado el texto, se tomarán las palabras de a una y se generará un diccionario, que pasará a ser parte de un modelo del estilo de Bag of words. El modelo difiere del Bag of words estándar en que en lugar de contar la frecuencia de cada palabra del diccionario por cada review, se estará contabilizando la frecuencia de aparición de dicha palabra en todas las reviews positivas, y por otro lado en todas las reviews negativas.

El resultado final del entrenamiento va a ser un Bag (diccionario) conteniendo como clave la palabra y como valor el índice a dos vectores, los cuales van a tener la frecuencia con la que aparece esa palabra tanto en las reviews positivas como en las reviews negativas.

A continuación se presenta un ejemplo del modelo a utilizar:

Con un archivo de reviews de tipo

```
sentiment    review
"0"          "This film sucks."
"1"          "The movie was great."
"1"          "I liked this movie, it was great."
```

Se genera un diccionario, cuya clave es la palabra y su valor es el índice para ambos vectores de frecuencias:

```
{
   "movie": 4
   "great": 2
   "sucks": 1
   "film": 0
   "liked": 3
}
```

Y sus correspondientes vectores de frecuencias:

```
vector_positivo = [ 0, 0, 2, 1, 2 ]

vector_negativo = [ 1, 1, 0, 0, 0 ]
```

En el caso del perceptrón, no se utilizará el modelo de Bag of words ilustrado anteriormente. En su lugar, se procesará la línea por completo, guardando también el sentiment indicado en el archivo de entrenamiento. Esto se realiza para que el perceptrón pueda validar sus predicciones, de manera que pueda compararlas con el sentiment leído en un principio.

Además, la forma en la cual se procesan las palabras en el algoritmo de perceptrón (utilizando una función de hash), nos permite utilizar un bigram, que pensamos nos provee un mejor entendimiento del lenguaje natural, al tomar, por ejemplo, conjuntos de sustantivo y adjetivo.

Aporte del grupo:

La idea del grupo fue, para probar, empezar con un Bag of words simple en conjunto con el algoritmo 1 ya explicado. Luego de ver que se superó el 80% de aciertos, se pensó que además de solo perfeccionar dicho algoritmo, se podrían usar distintos algoritmos y combinarlos para obtener un mejor puntaje. Dicha inspiración salió de que en clase se dijo que los últimos algoritmos que actualmente estaban ganando todas las competencias consisten de eso mismo.
Las ideas que no están probadas pero prometen mucho son las de asignar pesos a las palabras en el Bag acorde a la frecuencia que tengan (Algoritmo 2), usar el archivo AFINN.txt para asignar pesos a las palabras, asignar pesos a los distintos algoritmos y perfeccionar el Perceptrón.

Bibliotecas:
Por ahora no se piensa utilizar ninguna librería externa.

Apéndice

Comentarios sobre el TP

En consideración del grupo, el trabajo práctico resultó de extrema fascinación. El problema planteado es muy diferente a los problemas que los integrantes venían enfrentando en materias anteriores. Ninguno del grupo tiene experiencia laboral previa en el ámbito de la informática, y nunca se había encontrado con un problema donde es más importante la investigación y desarrollo de la solución, que la programación en sí misma.
Causó mucho interés el tema de machine learning, ya que es un tema muy nuevo del cual queda mucho que investigar y aprender. Se considera como algo bueno que en la facultad se vean temas más actuales, y que no son tan cotidianos o se escuchen en todos lados.
Por otro lado, resultó interesante también la aplicación que tienen otras materias que no son del área para el TP (como por ejemplo Probabilidad y Estadística para nuestro algoritmo clasificador bayesiano), ya que muchas veces uno se pregunta dónde se pueden llegar a utilizar ciertos conceptos, y es aquí donde vemos una aplicación concreta.

Conocimientos previos y expectativas

Uno de los integrantes del grupo, Santiago, tuvo contacto con algunos de los conceptos que se utilizan para resolver este tipo de problemas, durante un curso de Machine Learning dictado en Ciudad Universitaria. Generó interés en el tipo de problemas (que nunca antes había visto), pero nunca llegó a resolver uno.

Pruebas realizadas y resultados obtenidos hasta el momento

A modo de prueba y para aprender a manejar el set de datos probando los algoritmos investigados, se tomaron distintos conjuntos de reviews del archivo labeledTrainData.tsv para entrenar los distintos modelos, y luego se tomaron otros para probar los algoritmos de clasificación. Se decidió utilizar las reviews etiquetadas con sentiments para poder tener una idea de la cantidad de aciertos con la que se estaba trabajando.

Los resultados que se obtuvieron trabajando en Python, utilizando 20.000 reviews como entrenamiento y probando con las restantes 5.000, fueron los siguientes:

MÁS MENOS UNO
- Total de palabras: 67.690. Hay 4.151 / 5.000 aciertos. Porcentaje: 0.8302
- Usando un Lemmatizer. Total de palabras: 60.605. Baja la cantidad de aciertos: 4.121 / 5.000. Porcentaje: 0.8242
- Usando Porter Stemming. Total de palabras: 37.649. Baja la cantidad de aciertos: 4.045 / 5.000. Porcentaje: 0.809

NAIVE BAYES
- Total de palabras: 67.690. Hay 4.227 / 5.000 aciertos. Porcentaje: 0.8454. Tiempo: 122 minutos 34.642segundos

PERCEPTRÓN
- Hay 4.456 / 5.000 aciertos. Porcentaje: 0.8912. Tiempo: 4 minutos

Ideas a probar y trabajar

Como ideas complementarias a las ya expresadas anteriormente, también se pensó en utilizar el archivo AFINN-111.txt para pesar las palabras de las reviews, con el objetivo de mejorar el modelo entrenado. Estas palabras podrán también ser almacenadas en el Bag (para todos aquellos algoritmos que lo utilicen) y el peso será almacenado en el vector correspondiente.

Otra forma en la cual se pueden cargar las frecuencias es dándoles pesos según el tamaño de la review en la que aparecen. Una palabra no pesa lo mismo si aparece en un review de 1000 palabras o en una de 10 palabras.
También en vez de contar cuantas veces aparece la palabra en todas las reviews se puede contar en cuantas reviews aparecen.

Otra idea a probar es el uso de bigramas en aquellos algoritmos que utilicen un Bag. De esta forma, no solo se estarán utilizando las palabras individuales, sino que también se utilizarán palabras tomadas de a dos para comprobar si de esta manera se logra aumentar el nivel de precisión de los algoritmos.