Herramienta para analizar datos de un curso abierto masivo en línea

Alberto Pacheco González1, Isidro Robledo, René Cruz, Elisabet González

División de Estudios de Posgrado e Investigación. Instituto Tecnológico de Chihuahua. Av. Tecnológico No. 2909. Colonia 10 de Mayo, Chihuahua, Chih. C.P. 31310, México

Abstract

La gran cantidad de participantes en los cursos abiertos masivos en línea, implica una gestión explosiva de datos, cuyo análisis puede contribuir a mejorar de muchas formas un curso o el nivel de desempeño un participante. Se presenta el diseño e implementación de una herramienta de código abierto para analizar grandes cantidades de datos generados por 13,106 participantes de un curso abierto masivo en línea. Se detallan las características específicas del proceso de análisis de los datos, aplicando para ello una metodología de desarrollo de software que combina las bondades del desarrollo iterativo, orientado a objetos y de prototipos. La herramienta se implementó en Python y se validó a través del análisis de las respuestas de una encuesta de retroalimentación del curso compuesta por 37 preguntas que fue aplicada por medio de Formularios de Google a 1,867 participantes del curso en línea. La herramienta propuesta ofrece, entre otras ventajas, la capacidad de procesar cualquier cantidad de filas de una tabla al optimizar el manejo de memoria y limitar la granularidad del procesamiento a una sola fila hasta agotar todas las filas de una sola columna por cada ciclo del análisis. El diseño se generaliza para procesar distintos tipos de respuestas reusando gran parte del código, logrando reducir el tamaño de la herramienta de 650 a 250 líneas de código, al aplicar técnicas de programación orientada a objetos y otras cualidades del lenguaje Python que derivaron en un código muy compacto, entendible y genérico que puede ser extendido fácilmente para especificar distintos tipos de respuestas y analíticos. El caso de prueba presentado permite indirectamente validar las cualidades del lenguaje de programación Python para atacar problemas relacionados con el procesamiento de grandes datos; las pruebas realizadas arrojaron resultados muy positivos, encontrándose también ciertas limitaciones y se elaboran algunas recomendaciones para desarrollos a futuro de este tipo.

Palabras Clave: Big data; Data Analytics; Data Mining; MOOC; e-Learning; Analytics in Education

1. Introducción

Las decenas de miles de participantes de un típico curso abierto masivo en línea, *MOOC* por sus siglas en inglés, implica en su conjunto, una generación y gestión explosiva de datos, cuyo análisis puede contribuir a mejorar de muchas formas un curso en línea *per se*, incluyendo la obtención de métricas y sugerencias en tiempo real respecto al desempeño del participante del curso. Es por ello que el análisis de grandes datos (*big data*), la minería de datos (*data mining*) y la analítica del aprendizaje (*learning analytics*) están incidiendo de manera cada vez más notoria y decisiva dentro del estado del arte y futuro de la educación en la nube y los MOOCs en particular [1- 4]. Desde luego la analítica del aprendizaje es muy joven y se encuentra aún en su etapa formativa y experimental. En su primera conferencia internacional celebrada en 2011, fue definida como *"la medición, recolección, análisis y reporte de los datos acerca de los estudiantes y su respectivo contexto, con el propósito de entender y optimizar su aprendizaje y el entorno donde dicho aprendizaje sucede"* [1].

* *E-mail:* apacheco@itchihuahua.edu.mx

El problema de interés del presente trabajo consistió en atender la necesidad de procesar una gran cantidad de información generada por los participantes del curso en línea "Entendiendo el Cálculo Integral" perteneciente a la primera generación de MOOCs del Tecnológico Nacional de México (TecNM), ofertados todos ellos, a través de la plataforma MéxicoX[2], hospedada en un servidor basado en Open edX y gestionada por la SEP/DGTVE. La plataforma MéxicoX fue galardonada recientemente con el premio WSIS 2016[3]. El MOOC impartido del 6 de marzo al 7 de mayo de 2016, contó con una matrícula inicial de 13,106 estudiantes y alcanzó un notable índice de aprobación del 21% [5]. El objetivo del presente trabajo consistió en desarrollar una herramienta que fuera de utilidad para procesar la gran cantidad de datos generados por los participantes del MOOC en cuestión, esto con el objeto de elaborar los reportes oficiales y además, realizar un trabajo de investigación reportado en [5]. Este trabajo consistió en analizar una comunidad de estudiantes que participó en una red social (Twitter) asociada a la impartición del MOOC. Al finalizar el MOOC se comparon los perfiles de los tuiteros más participativos (EMAT) para determinar si dichos estudiantes obtuvieron un rendimiento similar al de un estudiante promedio del MOOC. Gracias al análisis de los grandes datos del MOOC y Twitter fue posible encontrar una mayor incidencia en EMAT que en MOOC, de estudiantes de alto y bajo rendimiento. Además se confirmó una distribución asimétrica (*fat tailed distribution*) en el nivel de participación del MOOC [5]. Dicho fenómeno también se presentó dentro del patrón de participación de los estudiantes de la comunidad de Twitter.

Entre las diversas fuentes de información generadas a partir del MOOC [5], se usará una sola fuente para reportar la validación de la herramienta dentro del presente trabajo. Dicho conjunto de datos (*dataset*) corresponde a una encuesta de satisfacción y retroalimentación del curso que se aplicó al final a los participantes del MOOC. El cuestionario constó de 37 preguntas (30 preguntas cerradas y 7 preguntas abiertas) [5]. Dicho cuestionario se aplicó vía Formularios de Google durante la primera semana de mayo de 2016 y fue contestada por 1,867 participantes. Desafortunadamente, las herramientas de Google (Google Cloud, BigQuery, Google Drive API y Google Apps Script) por estar más orientadas a mercadotecnia, comercio y negocios (servicios de web-click streams, web logs, AdSense, AdMob, Doubleclick), no fue posible localizar un API de Google que soportara adecuadamente la analítica de datos requerida tanto para los formularios como en hojas electrónicas generadas a partir de la encuesta. En delante dentro de este reporte, nos referiremos a la fuente de datos seleccionada como "encuesta final" y para su representación nos basaremos en un modelo de datos tipo *tabla* compuesta por *filas* (conjunto de respuestas correspondientes a cada participante) y *columnas* (respuestas obtenidas para cada pregunta del cuestionario).

2. Objetivo y especificaciones generales

El objetivo del presente trabajo consiste en diseñar e implementar una herramienta para procesar hojas de cálculo en formato CSV (*comma separated values*). Dicha herramienta debe ser lo suficiente versátil y genérica (no necesariamente limitada a una encuesta para un MOOC) para cubrir una serie de requerimientos con un mínimo de cambios en la codificación de cada analítico requerido. Para efectos del presente reporte se presenta código a manera de caso de estudio en Python para validar con los datos de la tabla de la encuesta final del MOOC. El esquema de procesamiento de datos debe estar automatizado en el mayor grado posible, evitando en lo posible la necesidad de la intervención manual por parte del usuario. La hoja de cálculo que conforma los datos de entrada tiene un formato CSV de texto plano y caracteres codificados en UTF-8. El resultado de la encuesta consiste en una tabla donde cada columna representa una pregunta y con excepción de la primera fila reservada para el encabezado de la tabla (donde cada columna contiene la leyenda de cada pregunta), tenemos que cada fila de la tabla contiene todas las respuestas de un determinado individuo encuestado separadas por comas, donde una respuesta/columna puede estar vacía, tener

[2] Plataforma MéxicoX disponible en: http://mx.televisioneducativa.gob.mx
[3] Noticia publicada en: http://bit.ly/2by4QA1

un dato o diversas respuestas separadas por punto-y-coma. El tipo de analíticos usado se limitó al conteo de frecuencias para cada tipo de respuesta, ya sea de opción múltiple, entrada numérica, una opción de texto o una respuesta abierta redactada en texto. Dichas respuestas las podemos clasificar para su procesamiento en cualquiera de los siguientes formatos: un formato numérico, una fecha o una cadena de texto, esta última puede ser alguna opción que corresponde a un texto preestablecido o un conjunto de palabras a buscar dentro del texto de la respuesta. El resultado del procesamiento se limita a la impresión (en texto plano) de cada analítico que consiste en el conteo de frecuencias ordenado de manera descendente (de forma numérica y porcentual).

3. Metodología

Para lograr el objetivo del presente trabajo fue necesario combinar diversas metodologías de desarrollo de software, específicamente: el desarrollo de prototipos rápidos, el diseño iterativo y el diseño orientado a objetos. En el presente estudio, por limitacionesde espacio, se presenta el código resultante de la última iteración del proceso iterativo analiza-diseña-implementa-y-valida. Dicha iteración corresponde a la aplicación de los principios de la programación orientada a objetos. Para el desarrollo basado en estas metodologías, fue necesario primero identificar los distintos tipos de reportes. En esta etapa de análisis se identificaron los siguientes casos o tipos de reportes: a) conteo de frecuencias simple; b) conteo de frecuencias de varios niveles/respuestas; c) un caso especial para respuestas tipo fecha; d) un caso especial para búsqueda de términos específicos dentro del texto de la respuesta. A nivel de diseño, implementación y pruebas se decidió separar el proceso en las siguientes etapas/ciclos: a) primer prototipo consistió en una prueba de concepto para cada caso, dedicando una iteración independiente para cada caso, comenzando por el caso más simple; b) generar los primeros reportes al nivel más básico posible; c) realizar transformaciones de la tabla de datos de entrada: sustituir la primer fila y eliminar columnas innecesarias; d) primera refactorización de código y su correspondiente documentación de código; e) insertar validaciones y manejo de errores (código robusto); f) identificar patrones y aplicar el diseño orientado a objetos; g) segunda refactorización, documentación y optimización de código/funcionalidad.

4. Diseño e implementación de la herramienta

Para realizar el estudio de comportamiento de los participantes del MOOC [5], fue necesario integrar y procesar información de diversas fuentes:

1) Información de la plataforma edX sobre el rendimiento académico de los participantes del MOOC.
2) Información proveniente de Google Forms de la encuesta aplicada a los participantes al finalizar el MOOC.
3) Información proveniente de los perfiles de Twitter de los participantes del MOOC que realizaron actividades y dinámicas durante la duración del curso.

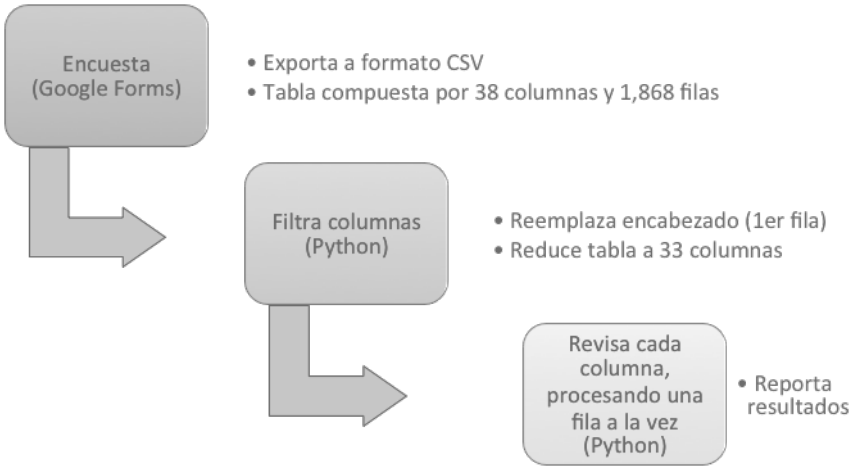

Fig. 1. Proceso por lotes seguido para efectuar el tratamiento de la información.

4.1. Estrategia para procesar grandes cantidades de datos.

Para procesar la gran cantidad y diversidad de datos del MOOC, resultó impráctico usar hojas de cálculo en Excel, razón por la cual se buscó otra alternativa que consistió en desarrollar una serie de scripts en Python respetando todas las especificaciones mencionadas arriba, para ello se plantearon las siguientes estrategias (fig. 1):

1. Exportar manualmente a formato CSV todos los datos almacenados en la nube de las 1,867 respuestas de la encuesta aplicada vía Google Forms. La tabla tiene 1,868 filas, donde la primer fila se reserva para el encabezado que contiene en cada columna el nombre de cada pregunta y tiene además 38 columnas incluyendo en la primer columna una estampa de tiempo y cada una de las 37 columnas restantes corresponde a una respuesta por parte del participante a cada una de las preguntas de la encuesta (una columna por cada respuesta).
2. Desde un punto de vista de diseño, se tomó la decisión de optimizar y modificar la hoja de cálculo original (*dataset*) por las siguientes razones:
    a. Las dimensiones de la tabla original (38 columnas por 1,868 filas) pueden repercutir de manera importante en la memoria principal requerida y su respectiva velocidad de procesamiento.
    b. Existen algunas columnas de la tabla que no son necesarias para obtener los analíticos y reportes.
    c. Al brindar la capacidad de seleccionar y reducir el conjunto de datos (*dataset*), podemos brindar una mayor versatilidad y aplicabilidad para la herramienta desarrollada.
3. Se considera importante establecer un mecanismo de indexación para la tabla de datos, cuyo nivel de abstracción sea lo suficiente genérico, expresivo y eficiente. Esto permitirá recorrer de manera eficiente la colección de datos y ocultar la complejidad del modelo de datos y sus detalles de implementación, logrando también maximizar la aplicabilidad y portabilidad de la herramienta propuesta.
4. La última estrategia consistió en crear un script independiente para procesar cada columna de la tabla. Cada uno de estos scripts a su vez procesa mediante un ciclo cada fila de la tabla, minimizando así los

requerimientos de espacio en memoria. Cabe mencionar que este mismo algoritmo se aplica de forma recursiva para procesar aquellas preguntas que cuentan con más de una respuesta.

A continuación se describe la implementación de cada una de dichas etapas, así como su versión orientada a objetos que permitió identificar los patrones comunes a todos los scripts, para establecer un solo código genérico que sea reutilizable y adaptable para considerar otras posibles variantes, e.g. distintos tipos de respuestas.

4.2. Primera etapa del proceso: Archivo en formato CSV

Como primer paso del proceso, desde la página de Google Forms donde se elaboró y aplicó la encuesta, se descargan directamente las respuestas como un archivo de texto en formato CSV (fig. 2).

Fig. 2. Operación de descarga de las respuestas de la encuesta del MOOC desde Google Forms.

4.3. Segunda etapa del proceso: Reducción del conjunto de datos (dataset)

El archivo de entrada `encuestaGoogle.csv` contiene todas las respuestas de la encuesta. Dentro del script `reduce.py` (fig. 3) el usuario especifica en `fila1` (línea 1) el nombre simbólico (una palabra de texto) para identificar cada columna y además se especifica en `excluye` los nombres de las columnas que serán excluidas. Enseguida se obtienen los índices de cada columna vía `map()` y se reordenan de forma que puedan ser borrados de forma segura del encabezado `fila1`. Se lee el archivo de entrada y se procesa cada fila (líneas 9-16). La primera fila se reemplaza por el nuevo encabezado `fila1` (11-13). Luego, de acuerdo a los índices de las columnas de `elimCol`, se eliminan de la fila siguiendo el orden adecuado (líneas 14-15). Al final de cada iteración se graba cada `fila` procesada en el archivo de salida (línea 16). A continuación presentamos el script `reduce.py` disponible en GitHub (http://github.com/albertochiwas), omitiendo por claridad, la clase `FileCSV` y las claúsulas para manejo de errores:

```
1: fila1   = ['hora','nombre','sexo','edad','correo'... ] # Encabezado
2: excluye = ['hora','nombre','escuela','ciudad'] # Columnas a eliminar
3: elimCol = sorted(map(lambda x: fila1.index(x),excluye),reverse=True)
4: for i in elimCol: # Para cada indice-columna...
5:     fila1.pop(i)  #    elimina dicha columna del encabezado
6: dataset = FileCSV.reader('encuestaGoogle.csv') # Tabla original
7: salida  = FileCSV.writer('encuesta.csv')       # Tabla modificada
```

```
 8: numFila = 0             # Contador de filas
 9: for fila in dataset:    # Procesa cada fila...
10:    numFila += 1         #    actualiza contador
11:    if numFila == 1:     #    si es la primera fila...
12:        salida.writerow(fila1) # se reemplaza con nuevo encabezado
13:        continue
14:    for col in elimCol:   # Para cada columna excluida...
15:      fila.pop(col)       #   eliminar dicha columna en fila actual
16:    salida.writerow(fila) # Graba fila actual en archivo de salida
```

Fig. 3. Python script '`reduce.py`' para modificar hoja de cálculo original: establece encabezado y elimina columnas.

4.4. Tercer etapa: Mecanismo de indexación

Al identificar cada columna por un nombre corto dentro del encabezado de la tabla (primer fila), se busca eliminar toda interdependencia entre la posición relativa de cada columna y su mecanismo de indexación, reemplazando para ello un índice numérico por un valor simbólico independiente de su posición. Esto evita además tener que reasignar nuevos índices numéricos a cada columna al momento de insertar o borrar columnas. Para realizar esta indexación simbólica efectuamos las siguientes operaciones:

1. Antes de procesar los datos, se obtiene la lista de llaves simbólicas de la primer fila del archivo `dataset`, esta lista servirá de base para indexar simbólicamente cualquier columna. La lista se consulta mediante:

```
columnas = dataset.next()  # Lista de llaves
```

2. Para indexar una determinada columna obtenemos, como paso intermedio, su índice numérico `col` a partir de su llave simbólica `nomCol` mediante:

```
col = columnas.index(nomCol) # Indice numérico
```

3. Dada una `fila` proveniente de la tabla `dataset`, podemos extraer el contenido de una columna `col` para dicha fila usando:

```
dato = fila[col]      #   lee su columna col
```

Podemos abstraer aún más dicho mecanismo de indexación simbólica hasta lograr representarlo de forma análoga a como se representa y accede un arreglo bidimensional:

```
dato = tabla[fila,columna]
```

Para ello basta refinar el procedimiento descrito anteriormente y explotar algunas de las bondades que ofrece un lenguaje de programación como Python. Para generar y asociar de manera automática los índices numéricos a su contraparte simbólica podemos auxiliarnos de un diccionario `mapCol` para mapear el nombre simbólico a un índice numérico de una columna. Para ello asociamos a cada identificador simbólico de columna el índice que corresponda a su posición relativa dentro de la lista `columnas`, registrando dicho par llave/valor dentro del diccionario auxiliar `mapCol`. Este diccionario puede generarse en Python convirtiendo la lista de tuplas que genera la primitiva **`zip`**`(list1,list2)`, donde cada tupla relaciona un elemento de `lista1` con un elemento de `lista2`; donde

`lista1` contiene los nombres de las columnas y `lista2` contiene los índices numéricos generados mediante `range()`:

```
mapCol = dict(zip(columnas,range(0,len(columnas))))
```

Esta forma de indexar simbólicamente se descompone a su vez en dos operaciones de indexación:

1. Indexamos el diccionario auxiliar `mapCol` usando como llave el nombre de la columna `nomCol` (índice simbólico) para obtener su índice numérico `col`

```
col  = mapCol[nomCol]
```

2. Enseguida obtenemos de la lista `fila` el contenido de la columna buscada mediante el índice `col`:

```
dato = fila[col]
```

En Python podemos encapsular estas operaciones usando el método de clase `__getitem__()` reservado para sobrecargar el operador de indexación `[]`. Podemos de esta forma indexar la hoja de cálculo como una matriz usando dos índices, i.e. `tabla[fila,columna]`, usamos lo siguiente:

```
def __getitem__(self, tupla):    # Operador de indexación []
      fila, nomCol = tupla       # Extrae ambos índices
      col = self.mapCol[nomCol]  # Indexa diccionario auxiliar
      return fila[col]           # Regresa valor en dicha columna
```

Dentro de dicha clase podemos acceder el contenido de una columna `nomCol` para la `fila` actual mediante:

```
dato = tabla[fila,nomCol]  # Indexación tabla[fila,col]
```

Esta última sentencia es idéntica a como accedemos un arreglo bidimensional, pero en realidad es una llamada al método `__getitem__` expuesto arriba. Cabe mencionar que el índice para la `fila` es, en realidad, un objeto (lista de columnas) y el índice para la columna `nomCol` es un texto con nombre de la columna.

4.5. Última etapa del proceso: Procesar datos

Una vez que se obtiene la tabla a partir de un archivo en formato CSV (etapa 1), se modifica dicha tabla, tanto su encabezado para identificar cada columna, como eliminando las columnas innecesarias (etapa 2), se establece un mecanismo de indexación apropiado para recorrer el conjunto de datos (etapa 3), se procede finalmente al procesamiento de los datos para obtener el conteo de frecuencias de las respuestas para cada pregunta, i.e. todas las filas de cada columna, cuyo algoritmo básico representado en pseudocódigo queda como sigue:

```
1: for col in tabla.columnas:            # Por cada pregunta...
2:    resultados = Resultados()
3:    for fila in tabla.filas:           # Por cada usuario...
4:        respuestas = tabla[fila,col]
5:        for resp in respuestas:        # Para cada respuesta...
```

```
6:            resultados.frec[resp] += 1 #    actualiza conteo
7:    reporta(resultados)
```

Fig. 4. Algoritmo principal para etapa de procesamiento de cada pregunta (columna).

El ciclo externo del algoritmo (fig. 4) recorre la tabla por columnas (líneas 1-7), donde cada columna representa una pregunta y el conjunto de respuestas para dicha pregunta. Una instancia de la clase `Resultados` (línea 2) permite entre otras cosas, actualizar el conteo de frecuencias de las respuestas de cada pregunta (línea 6). Para ello, procesamos todas las filas de la tabla (líneas 3-6) accediendo a cada respuesta indexando simbólicamente el contenido de la tabla para la fila y columna actual (línea 4). Dado que en ocasiones podemos tener más de una respuesta en una pregunta, requerimos un último ciclo (líneas 5-6) para procesar todas las respuestas diferentes que haya indicado un usuario para dicha pregunta (líneas 4-6). Para cada respuesta, usamos su propio valor para indexar y actualizar el contador de frecuencias `resultados.frec` (línea 6). Al final del recorrido de todas las filas de una columna se reportan sus resultados correspondientes (línea 7).

5. Discusión y Resultados

En la práctica, al momento de implementar el algoritmo propuesto (fig. 4) se presentaron diversas dificultades. En primer lugar, Google Forms ofrece distintos tipos de respuestas: numéricas, fechas, textos con un contenido predefinido o un texto arbitrario escrito por un usuario. En algunos casos no resulta conveniente registrar la respuesta completa como llave del diccionario para el conteo de frecuencias `Resultados.frec` (línea 6, fig. 4). Por ejemplo, para los comentarios finales, su análisis requirió efectuar una búsqueda de determinadas frases o patrones de texto para contar el número de comentarios positivos o negativos. La fecha de nacimiento fue otro caso que proceso un texto de entrada arbitrario para estimar la edad del participante y clasificar además por rango de edades. Esto se solucionó derivando a partir de la superclase `Resultado` una subclase `Edades` para procesar por rango de edades y otra para subclase `Comentarios` para el análisis de textos para clasificar comentarios (fig. 5). A su vez, dichas subclase redefinen el método polimórfico `update()` para analizar las respuestas de acuerdo a sus propios criterios y realizar el conteo de frecuencias de manera particular (fig. 5):

Fig. 5. Arbol de clases para los distintos tipos de respuestas.

Tomando lo anterior en cuenta, fue necesario adecuar el algoritmo original (fig. 4) para atender cada tipo de respuesta (fig. 5). El nuevo algoritmo (fig. 6) es capaz de seleccionar la subclase de `Resultados` más adecuada para cada columna (líneas 2-7 de fig. 6):

```
1: for col in tabla.columnas:
```

```
2:    if col == 'edad':
3:       resultados = Edades()
4:    elif col == 'comentario':
5:       resultados = Comentarios()
6:    else:
7:       resultados = Resultados()
8:    tabla.analize(col,resultados)
9:    reporta(resultados)
```

Fig. 6. Algoritmo principal en su versión final.

El nuevo método `analize()` encapsula el procesamiento de las filas de una determinada columna (líneas 3-6 de fig. 4) y recibe como parámetro una instancia de alguna subclase de `Comentarios` (línea 8 en fig. 6), para analizar todas las respuestas del mismo tipo (filas) para una determinada pregunta (columna). Finalmente, la clase `Encuesta` (fig. 7) implementa todas las etapas del proceso: lee y modifica la tabla vía el constructor `__init__()`, indexa la tabla vía `__getitem__()` y procesa cada pregunta vía el método `analize()`. La implementación de la clase (fig. 7) posee un alto nivel de abstracción. El método `analize()` es independiente de la implementación y tipo de estructura de datos (usando un iterador universal - línea 7), indexa la tabla como un arreglo normal (línea 8) y usa un método polimórfico para analizar todo tipo de respuesta en base a la clase que pertenezca el objeto `resultados` recibido como parámetro del método `analize()`.

```
 1: class Encuesta:
 2:    def __init__(...):    # constructor: lee y simplifica tabla
 3:    ...
 4:    def __getitem__(...): # indexación simbólica vía operador []
 5:    ...
 6:    def analize(self, col, resultados):   # Analiza por tipo de respuesta
 7:        for fila in self.filas:           # Por cada fila...
 8:            respuestas = self[fila,col]   #   Indexa tabla
 9:            for resp in respuestas:       #   Por cada respuesta...
10:                resultados.update(resp)   #    Procesa según su tipo
```

Fig. 7. Clase desarrollada para procesar encuestas con un gran número de respuestas y tipos de preguntas.

Gracias a la metodología de desarrollo iterativo de prototipos fue posible generar los primeros scripts funcionales desde la primer sesión de desarrollo y lograr así analizar y entregar un primer reporte de la encuesta del curso MOOC en la misma semana que se aplicó la encuesta. Puesto que se espera que los cursos del TecNM sigan ofertándose, se realizó una nueva iteración de desarrollo para refactorizar y mejorar el reuso del código aplicando la programación orientada a objetos. Esta última versión permitió generar los analíticos de todas las preguntas ejecutando un sólo script, mismo que tardó solo 1.24 segundos en procesar las 33 preguntas de los 1,867 participantes que contestaron la encuesta de retroalimentación del MOOC (fig. 8).

En general, como se ha demostrado en el presente artículo, el código de Python es muy legible, estructurado y expresivo. La implementación completa de la herramienta usando este lenguaje fue relativamente simple (3 iteraciones en 3 sesiones de 3 horas de desarrollo cada una), más una última sesión de 4 horas para refactorizar los 22 scripts que analizan por separado cada pregunta (650 líneas de código en total) y desarrollar las clases de la

versión final contenida en sólo 3 scripts: la utilería `reduce.py`, la librería `encuesta.py` y el reporte `enc-reporte.py` (casi 250 líneas de código en total, representando una reducción de 62% del código anterior). También fue posible adaptar en muy poco tiempo (menos de una hora) dicha herramienta para procesar archivos distintos, que no provenían de los Formularios de Google, como fue el caso de los resultados finales de las evaluaciones de los participantes del curso MOOC obtenidos directamente de la plataforma Open edX y lás métricas obtenidas de la comunidad de usuarios de Twitter.

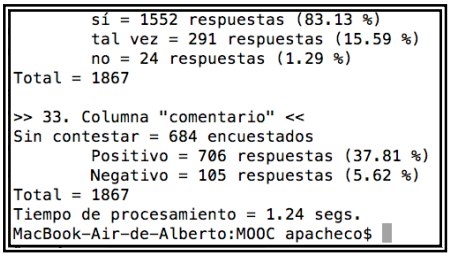

Fig. 8. Reporte generado con la herramienta al momento de analizar la encuesta del MOOC.

6. Conclusiones y trabajo futuro

La creciente demanda y aceptación de los cursos en línea masivos y abiertos (MOOC), donde un solo curso puede atender decenas de miles de participantes, brinda sentido al hecho de aplicar el análisis de grandes datos y la minería de datos (*big data & data mining*), así como el hecho de almacenar y procesar datos en la nube (*cloud computing*) y realizar una analítica del aprendizaje en tiempo real (*learning analytics*) para adaptar de manera inmediata las prestaciones y elementos de un MOOC.

Gracias a la herramienta de código abierto desarrollada (scripts de Python) fue posible procesar toda la información de manera simple, rápida, robusta y eficiente (fig. 8) obteniendo así los analíticos del curso MOOC que sirvieron de base para generar reportes y estudios como se reportan en [5]. De otra manera, de no haber contado con dicha herramienta, el análisis de los datos usando hojas de cálculo hubiese sido un proceso tedioso, ineficiente, tardado y sujeto a errores humanos.

A futuro consideramos interesante adaptar y aplicar la presente herramienta para analizar en tiempo real el avance mismo de los participantes del MOOC (*learning analytics*), tarea que puede llevarse a cabo, sin mayor problema, si los datos de la plataforma se exportan en formato CSV. Otra área de oportunidad lo constituye, la generación automática de reportes y gráficos, ya que en la práctica, en nuestro caso, fue la tarea que consumió más tiempo, por lo que se publicará también un subproyecto de código abierto en GitHub[4], abierto a la comunidad general, tanto para mejorar la herramienta actual como el componente para visualizar reportes estadísticos usando otras herramientas de Python, tal como Jupyter Notebook[5].

El diseño de la herramienta y su implementación en Python fue adecuado para manipular archivos de gran tamaño considerando también: a) la eliminación de las columnas no requeridas; b) procesar en memoria solo una fila de la tabla a la vez, consumiendo menos del 0.05% de memoria respecto al tamaño de los archivos de entrada para la

[4] Código disponible en: http://github.com/albertochiwas/moocsv
[5] Jupyter Notebook disponible en: http://jupyter.org

encuesta con 1,867 registros con 33 columnas y las evaluaciones de los 13,106 participantes del MOOC respectivamente. Esta técnica permite procesar un archivo con un número ilimitado de filas, lo cual constrasta con el método tradicional de procesamiento de hojas de cálculo que necesita consultar en memoria el archivo completo.

El desarrollo de la presente herramienta ha permitido confirmar que el lenguaje de programación Python resulta apropiado para procesar grandes cantidades de datos. En nuestro caso solo fueron requeridos los elementos nativos del lenguajes, no fue necesario usar librerías avanzadas, costosas y especializadas, que también existen para este lenguaje, como es el caso de la librería NumPy. Otra ventaja de Python es que trabaja en todas las principales plataformas, tanto del lado del cliente como del lado del servidor, esto puede resultar atractivo para desarrollar a futuro las capacidades de la presente herramienta como un servicio en la nube, como por ejemplo, la herramienta para procesamiento concurrente más reciente de Intel [6], anunciada como tecnología de punta para afrontar los retos de procesamiento de grandes datos (*big data*) prometiendo así un futuro más promisiorio en esta área para el lenguaje de programación Python.

Referencias

1) Long PD, Siemens G. Penetrating the Fog: Analytics in Learning and Education. *EDUCAUSE Review* 2011; 46(5): 31–40. Disponible en: http://bit.ly/2blK8Y8
2) L. Johnson L, et. al. The NMC Horizon Report: 2013 Higher Education Edition. *NMC Horizon* 2013: 1–40.
3) Kay J, et. al. MOOCs: So Many Learners, So Much Potential ... *IEEE Intelligent Systems*, 2013; 28(3): 70–77.
4) Ferguson R. Learning Analytics: Drivers, Developments and Challenges. *Int. J. Technol. Enhanc. Learn.* 2012; 4(5/6): 304–317.
5) Pacheco A, Alvarez VC. Analizando el comportamiento del rendimiento académico de los participantes de un MOOC desde la perspectiva de una red social. En: Universidad Alicante/EduTec, *Educación y Tecnología. Propuestas desde la investigación y la innovación educativa*. Editorial Octaedro. ISBN: 978-84-9921-846-5, 2016 (publicación aceptada).
6) Intel, The Parallel Universe: Supercharging Python for Open Data Science. *Intel White Paper*. July 2016; Issue 25.