Recuperación de Información y Recomendaciones en la Web 2014

Proyecto Final

Tema:

Sugerencias de vendedores en sitios de e-commerce

Grupo 02
Integrantes:
4463446-7 Cecilia Barreiro
4538142-1 Verónica Gamarra
4257123-1 Nicolás García

Contenido

Introducción

Dado el auge de los sitios de venta por internet y que estos son cada vez más utilizados por las personas, vimos la oportunidad de desarrollar un sistema que ayude a los compradores a la hora de realizar una compra.

En la mayoría de los sitios de e-commerce como Mercado Libre, OLX u otros de mayor tamaño como Amazon o AliExpress los compradores luego de concretar su compra pueden puntuar y realizar comentarios sobre los vendedores. Dichos comentarios sirven para informar a los demás compradores sobre el vendedor, estos dan información por ejemplo sobre la puntualidad, calidad del producto y de la entrega, etc.

Los comentarios brindan información sobre las distintas compras realizadas por los clientes pero no dan información unificada y global sobre un vendedor.

El objetivo de nuestro proyecto se centra en brindar información global sobre los vendedores a partir del análisis lexicográfico de los comentarios realizados por los clientes, sin importar la calificación que éstos tengan.

Herramientas

Web crawler: Scrapy 0.24

En etapas tempranas analizamos dos herramientas para web crawling. Estas fueron OpenWebSpider y Scrapy. La primera es una herramienta libre basada en Linux mientras la segunda se encuentra escrita en Python.
Al momento de evaluar cuál utilizar se analizó facilidad de instalación, documentación disponible, lenguaje utilizado y adaptación a nuestro proyecto.
OpenWebSpider presentó algunas dificultades durante la instalación, mientras que Scrapy por el contrario implicaba la ejecución de un comando para su instalación, considerando que existen varios paquetes y programas previamente instalados (como el *pip* o *Python* en su última versión) con los cuales algunos de los integrantes ya contaban. También se analizaron ventajas como la agilidad y el lenguaje de programación (Python), con el cual ya nos encontramos familiarizados, También se investigó que la herramienta no es muy escalable, pero esto no tiene demasiada relevancia en nuestro trabajo ya que este implica analizar solo una categoría de un sitio en particular.

Python 2.7

Debido a que varios de los integrantes del grupo ya contaba con experiencia en Python y el web crawler elegido está programado en este lenguaje, se optó por elegirlo como lenguaje para nuestro proyecto. Además de la facilidad en su uso, Python facilita la integración con

las otras herramientas y módulos utilizados, como *pyswip*, que permite acceder a valores cargados para ciertos lemas, desde un archivo Perl, u *openpyxl*, que permite convertir un archivo *csv* en *xlsx*.

Manejador de paquetes Python: PIP

PIP es una herramienta para instalar y manejar paquetes Python, que facilita notoriamente la generación del ambiente adecuado para la ejecución, ya que con pocos comandos se pueden instalar todas las herramientas necesarias.

Freeling 3.1

Freeling es una herramienta open-source para el procesamiento y análisis del lenguaje, diseñada para ser utilizada como una librería por los programas que la requieran. Entre las funciones que ofrece están la tokenización y tagging así como también el análisis morfológico, cálculo de probabilidades de que la etiqueta otorgada a esa palabra sea efectivamente correcta dentro del contexto de la frase.

La herramienta se utilizó a partir del comando Popen de la librería Subprocess de Python, ya que la misma no presenta una librería para importar y utilizar directamente. Debido a que la consulta de cada comentario llevaba un tiempo considerable es que se investigó otra forma que permitiera procesarlos a todos más eficientemente. Se investigó el modo servidor que ofrece la herramienta, el cual una vez inicializado procesa el comentario casi instantáneamente debido a que el proceso ya está levantado con sus librerías y funciones. Pero desgraciadamente, aunque se logró ejecutar este método, el servidor se cierra una vez procesado un comentario. Investigando esto un poco más, llegamos a la conclusión de que es algo normal bajo Windows, y al parecer en Linux esto no ocurre, pero los cambios a realizar, además de todo el esfuerzo que conlleva cambiar de plataforma, es que se considera una futura mejora a este sistema.

Propuesta de problema

Utilizando la herramienta Scrapy recorreremos varias páginas de una categoría y producto en particular de Mercado Libre, navegaremos hacia las páginas de cada vendedor de esos productos obteniendo los comentarios hechos por clientes sobre él.

Los comentarios obtenidos serán analizados lexicográficamente y a cada uno de ellos se le asignará una ponderación (peso) total en base a el valor asociado de cada una de sus palabras relevantes (no se incluyen palabras vacías “stop-words”, números, fechas ni símbolos, que no aportan al valor del comentario en su conjunto).

Finalmente se agruparán esos comentarios por vendedor, sumando las ponderaciones de cada comentario para cada vendedor, devolviendo a los 10 vendedores mejor ponderados.

Propuesta de solución

Para la creación del prototipo nos centramos en el sistema de ventas Mercado Libre, ya que no se requiere autenticación para el acceso a los datos. Dentro de este sistema nos centramos en el análisis de los vendedores de televisores.
En Mercado Libre los comentarios realizados por los compradores son hechos sobre los vendedores y no sobre el producto comprado.

La implementación puede ser dividida en dos etapas, en la primera se obtienen los comentarios y en la segunda se realiza el análisis de peso lexicográfico de los mismos.

Primera etapa
En esta etapa se utiliza la herramienta de web crawling Scrapy[1] para obtener los comentarios y puntuaciones realizadas por los usuarios a los vendedores de televisores.
Para esto necesitamos obtener con Scrapy todas las publicaciones de televisores a partir de la página "principal" de venta de televisores *http://electronica.mercadolibre.com.uy/televisores/*, esto se lleva a cabo obteniendo los links a todos los televisores de la página actual y el link a la siguiente página del paginado existente. Este proceso se repite hasta llegar a la última página guardando los links de los televisores.
Los links en Scrapy se obtienen utilizando expresiones regulares sobre los elementos href del código html de las páginas y sobre restricciones de ubicaciones de elementos. Esto fue necesario ya que la página brinda sugerencias de otro tipo de productos que no son relevantes para nuestro estudio.

A continuación, a partir de los televisores se obtienen los perfiles de los vendedores. Se constató que en algunas situaciones varios televisores son vendidos por un único vendedor, en este caso decidimos tener en cuenta solo una vez a el vendedor para clasificarlo.
En la página de los vendedores obtenemos la puntuación y la primer página de comentarios realizados por los compradores debido que era necesario limitar la cantidad de comentarios guardados para poder procesarlos en un tiempo razonable.

Finalmente los comentarios y puntuaciones son asociados a los vendedores y guardados en un archivo local en formato "csv".

Segunda etapa
En la segunda etapa se realiza el análisis de los comentarios utilizando la herramienta Freeling. Con dicha herramienta realizamos la desambiguación morfosintáctica y la lematización de todos los comentarios, también son filtradas las fechas, números y símbolos que no son caracteres(utilizando las etiquetas que

[1] http://scrapy.org/

provee Freeling). El siguiente filtro está conformado por las palabras vacías, o “stop words”, que se conocen como aquellas que no poseen un significado, como son los artículos, pronombres, preposiciones, etc, las cuales fueron descartadas luego del análisis gramatical y no tomadas en cuenta.

Para realizar el análisis mencionado vimos conveniente convertir el archivo obtenido en la primer etapa “csv” a “xlsx” debido que existe un módulo compatible con Python que facilita el manejo de este tipo de archivos.
Este archivo se redujo en la cantidad de vendedores para limitar la cantidad de comentarios a procesar, ya que es muy costoso en término de tiempo.

Contamos con un conjunto de palabras “positivas” y “negativas” que son las que darán el peso lexicográfico a los comentarios, el cual se obtiene a partir de un archivo Perl, donde para cada lema está asignado un valor numérico.

Como primera aproximación al análisis de los comentarios definimos que un comentarios es positivo si contiene palabras “positivas” y es negativo si contiene palabras “negativas”. Para este tipo de clasificación teníamos un problema a la hora de decidir si un comentario era positivo o negativo cuando contenía palabras “positivas” y “negativas”. Por este motivo decidimos dar puntaje a los comentarios.

Para puntuar los vendedores analizamos todos sus comentarios, por cada uno de los comentarios obtenemos una ponderación y finalmente obtenemos la puntuación del vendedor.

Para obtener la ponderación de un comentario realizamos el siguiente procedimiento, si una palabra “positiva” se encuentra en el mismo se le suma un valor fijo a la ponderación parcial y si una palabra “negativa” se encuentra en él, se le resta el mismo valor. En este prototipo los valores que utilizamos fueron 3 y -3.

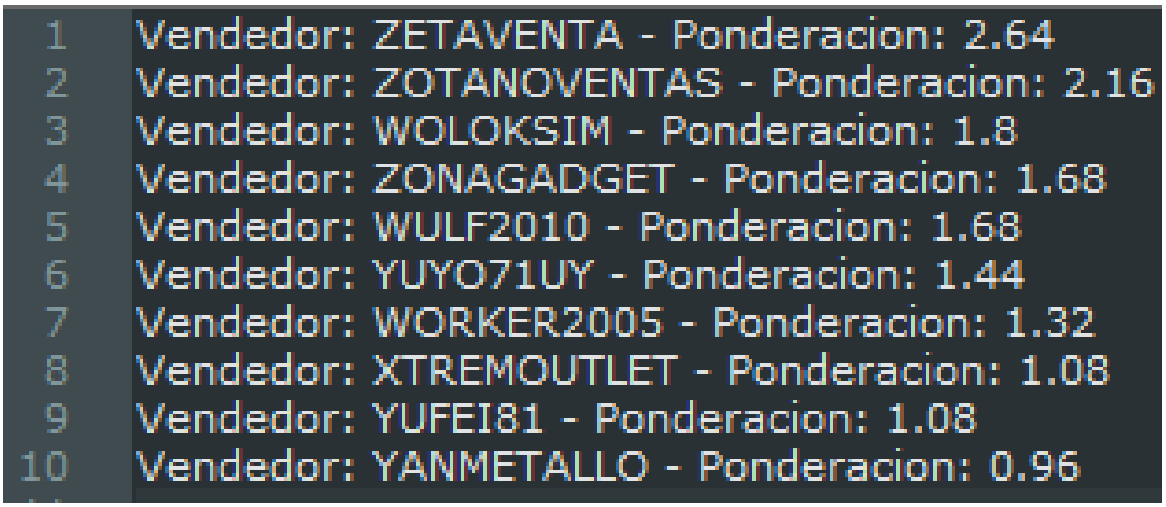
Finalmente se obtiene la puntuación de los vendedores sumando las ponderaciones de sus comentarios y dividiendo entre la cantidad total de los mismos. Cuanto mayor sea esta puntuación mejor clasificado se encuentra el vendedor.

Resultados

Luego de ejecutar la segunda etapa de nuestro proyecto pudimos obtener los datos de vendedores cuyos comentarios tuviesen un peso lexicográfico acorde para realizar una sugerencia eficaz al posible comprador.

Se obtuvo que el procesamiento de los comentarios tiene un tiempo de ejecución del orden de 2 horas para 10.000 comentarios, por lo cual disminuimos el tamaño del corpus a 375 comentarios correspondientes a 15 vendedores.
Observamos que varios comentarios tienen faltas de ortografía o están incorrectamente escritos, lo cual influye negativamente para la correcta asignación del peso lexicográfico de los comentarios. Por otro lado se observó que en algunos casos particulares que vendedores que poseían buenos comentarios no tenían una ponderación final acorde, por lo dicho anteriormente.

Se obtuvieron los 10 vendedores de mayor ponderación y para cada uno de ellos se devuelve además su puntaje promedio.



En el anexo se proporcionan los códigos fuente. Se adjunta un archivo leeme.txt con las indicaciones para poder compilar y ejecutar los diferentes proyectos.

Mejoras

- En una segunda etapa del prototipo deberíamos mejorar el conjunto de palabras “positivas” y “negativas”, esto es encontrar más y mejores palabras que indiquen si un comentario es positivo o negativo.
- A la hora de obtener una ponderación de comentario a partir de sus palabras deberíamos tomar en cuenta el contexto en que esta se encuentra. Ya que no es lo mismo el comentario “Lo recomiendo” que “No lo recomiendo”.
- Además se podría utilizar las puntuaciones realizadas a los vendedores para ponderar comentarios o vendedores y así ayudar a clasificar los vendedor.
- Para mejorar la técnica el análisis de los comentarios, se podría utilizar métodos de aprendizaje automático para la ponderación de comentarios de manera automática. Entrenar algún clasificador como por ejemplo NaiveBayesClassifier de Nltk para que realice la ponderación automática de los comentarios.
- Obtener una lista de mayor cantidad de volumen de datos a nivel de comentarios por vendedor.
- Aplicar técnicas para la mejora del tiempo del procesamiento del proyecto FreeLing.

Trabajos a futuro

- Ampliación de clasificador de vendedores en otras páginas de e-commerce como Amazon, OXL, etc. El análisis de los comentarios se realizará utilizando el mismo método, lo que cambia es la forma de obtener los comentarios y puntuaciones de vendedores ya que la estructura de las páginas (html) van a ser distintas.

- En otros sitios de compra se puede clasificar además del vendedor el producto. Por lo tanto una mejora importante sería poder realizar un análisis de las puntuaciones y comentarios de los productos con el objetivo de ayudar al comprador a la hora de decidir sobre un producto.

Conclusiones

A nivel de investigación, analizamos y trabajamos con diferentes herramientas que nos permitieron tener un acercamiento práctico al manejo de información en la web. Observamos que no existe un ambiente único donde contar con todas las funcionalidades integradas, por lo cual es necesario integrar diferentes tecnologías. En nuestro caso varias estaban desarrolladas en el mismo lenguaje, por lo que nos facilitó la instalación y utilización de las mismas.
Nos resultaron interesantes los resultados que se pueden obtener con las herramientas de web crawling. Se cuenta con un gran potencial al contar con esta información centralizada y desglosada con criterios previamente definidos. Y accedida posteriormente tanto desde una base de datos como en archivos de manipulación de datos como *.csv.

Por otro lado comprobamos la agilidad de la herramienta seleccionada para esta actividad ya que en escasos minutos se pudo obtener una gran cantidad de referencias analizadas.

Para el caso de las herramientas de procesamiento y análisis del lenguaje, se contaba con experiencia previa lo cual facilitó adaptar estos conceptos a nuestra realidad. De todos modos surgieron nuevos desafíos debido a la gran cantidad de datos que se manipularon. Se investigaron técnicas para poder agilizar los tiempos de procesamiento pero sin éxito debido a las características del sistema operativo de nuestro ambiente de desarrollo.
Fue necesario agregar palabras a la lista de términos con sus valores asignados, ya que varias palabras no estaban incluidas originalmente y podían tener influencia en el peso total del comentario.

Finalmente con relación al tema central de nuestro proyecto, pudimos realizar un prototipo con retornará los resultados esperados. Como se mencionó inicialmente, hoy día el mercado de e-commerce se encuentra en pleno crecimiento, el desarrollo de herramientas que manipule esta información generará un valor agregado a la usabilidad y experiencia de usuario en los sitios. No solo es importante contar con un buscador por productos eficaz, sino también brindarle al usuario un contexto de confiabilidad para así generar fidelidad.

Anexo

Código fuente tutorial_spider.py

```
from scrapy.contrib.linkextractors import LinkExtractor
from scrapy.contrib.spiders import CrawlSpider, Rule
from scrapy import log
from tutorial.items import CraigslistSampleItem


class MercadoLibreSpiderSpider(CrawlSpider):
	name = "mercadoLibreSpider"
	allowed_domains = ["mercadolibre.com.uy"]
	start_urls = [
		"http://electronica.mercadolibre.com.uy/televisores/"
	]
	rules = (
		## Identificacion paginado principal
	Rule(LinkExtractor(allow='http:\/\/electronica\.mercadolibre\.com\.uy\/tele
visores\/_Desde_[0-9]*', restrict_xpaths=('//li[@class="last-child"]'))),

		## Identificacion productos del panel principal
	Rule(LinkExtractor(allow='http:\/\/articulo\.mercadolibre\.com\.uy/[a-zA-
Z_/\-0-9+$]*', restrict_xpaths=('//ol[@id="searchResults"]'))),

		##Identificacion perfil a partir de la calificion del vendedor
	Rule(LinkExtractor(allow='http:\/\/perfil\.mercadolibre\.com\.uy/[a-zA-Z_\-
0-9+$]*', restrict_xpaths=('//a[@class="more-info"]')), callback='parse_access',
follow=False),
	)

	def parse_access(self, response):
		items = []
		item = CraigslistSampleItem()

		# Obtengo las reputaciones
		for sel in response.xpath("//nav[contains(@class, 'reputation-
filters')]"):
			valoracionesPos = sel.xpath('//a[re:test(@href,
".+type=positive.+")]/b/text()').extract()
			valoracionesNeutral = sel.xpath('//a[re:test(@href,
".+type=neutral.+")]/b/text()').extract()
			valoracionesNeg = sel.xpath('//a[re:test(@href,
".+type=negative.+")]/b/text()').extract()

			#Si la cantidad es cero obtengo a partir del span
			if len(valoracionesPos) == 0:
				item ["valoracionesPos"] =
sel.xpath("span[contains(i/@class, 'positive')]/b/text()").extract()
			else:
				item ["valoracionesPos"] = valoracionesPos[0]

			if len(valoracionesNeutral) == 0:
				item ["valoracionesNeutral"] =
sel.xpath("span[contains(i/@class, 'neutral')]/b/text()").extract()
			else:
				item ["valoracionesNeutral"] = valoracionesNeutral[0]

			if len(valoracionesNeg) == 0:
				item ["valoracionesNeg"]  =
sel.xpath("span[contains(i/@class, 'negative')]/b/text()").extract()
```

```
                    else:
                          item ["valoracionesNeg"] = valoracionesNeg[0]

             ### Obtengo comentarios ###
             lstComentarios = ''
             lstDatos = response.xpath('//i[@class="feedback-icon-
type"]/../text()')
             for i in range(len(lstDatos)):
                    lstComentarios += lstDatos[i].extract().replace(',',
'').replace('"', '').replace('\r', '').replace('\t', '').replace('\n', '') + ' |
'

             item ["comentario"] = lstComentarios

             #Obtengo usuario
             usr = list()
             for sel in response.xpath('//div[@class="reputation-user-data"]'):
                    usr = sel.xpath('h1/text()').extract()

             if len(usr) > 0:
                    item ["vendedor"] = usr[0]

             #Obtengo numero de paginado
             for sel in response.xpath('//li[@class="ch-pagination-current"]'):
                    item ["paginaComentario"] = sel.xpath('a/text()').extract()

             #Obtengo liknk de siguiente pagina de comentaios
             links = list()
             for sel in response.xpath('//li/a[@class="nextLink"]'):
                    links = sel.xpath('@href').extract()

             if len(links) > 0:
                    print "Link siguiente"
                    print links[0]

             items.append(item)
             return items
```

Código fuente proyecto FreeLing

```
import os
import csv
import operator
import re
import openpyxl # from https://pythonhosted.org/openpyxl/ or PyPI (e.g. via pip)
from openpyxl.cell import get_column_letter
from subprocess import Popen, PIPE
from pyswip import Prolog
import xlrd

base_cmd_freeling = ['analyzer', '-f', '%s/config/es.cfg' %
os.environ['FREELINGSHARE']]

def obtenerSalidaFreeling(entrada, outf, parametros):
       cmd =  base_cmd_freeling + ['--outf', outf] + parametros
       p = Popen(cmd, stdin=PIPE, stdout=PIPE, shell=False)

       # Freeling no soporta UTF-8
       salida_pipe = p.communicate(input=entrada.encode('utf-8'))[0].decode('utf-
8')
```

```
        return salida_pipe

def filtrar_palabra(etiqueta):
  return (etiqueta[0] == 'F' or etiqueta[0] == 'Z' or etiqueta[0] == 'W')

def cargar_stop_words():
        stop_words = {}
        with open("stop-words.txt") as f:
                data = f.readlines()

        for n, line in enumerate(data, 1):
                line_key = line.split('\n')[0]
                stop_words[line_key] = ""

        return stop_words

def convert_csv_to_xlsx():
  f = open(r'items.csv')
  wb =  openpyxl.Workbook()
  dest_filename = r"destino.xlsx"
  ws = wb.worksheets[0]
  ws.title = "Comentarios"
  reader = csv.reader(f)
  for row_index, row in enumerate(reader):
      for column_index, cell in enumerate(row):
          column_letter = get_column_letter((column_index + 1))
          ws.cell('%s%s'%(column_letter, (row_index + 1))).value = cell

  wb.save(filename = dest_filename)

def empty_elems(arr):
  all_empty = True;
  cant = len(arr);
  i=0;
  while (all_empty) and (i < cant):
    all_empty = len(arr[i]) == 0;
    i+=1;
  return all_empty;

class comentario:
  texto = ""
  pagina = 1
  vendedor = ""
  ponderacion = 0
  palabras = []
  texto_freeling = ""

  def __init__(self, texto, vendor, pagina_com):
              self.texto = texto
              self.pagina = pagina_com
              self.vendedor = vendor
              self.ponderacion = 0
              self.palabras = []
              self.text_freeling = ""

  def agregar_texto_freeling(self, texto_freeling):
              self.texto_freeling = texto_freeling

  def incrementar_ponderacion(self, valor):
              self.ponderacion += valor

  def agregar_palabras_comentario(self, palabra):
              self.palabras.append(palabra)
```

```
  def imprimir_texto_freeling(self):
              print 'Texto de freeling: ', self.texto_freeling

  def imprimir(self):
              print 'Comentario: ', unicode(self.texto).encode('utf-8')
              print 'Palabras: ', self.palabras
              print 'Vendedor: ', self.vendedor
              print 'Ponderacion', self.ponderacion

# PROGRAMA PRINCIPAL
try:

  # convert_csv_to_xlsx()
  # Se cargan las palabras vacias
  stop_words = cargar_stop_words()
  prolog = Prolog()
  prolog.consult('listasElementosPonderados.pl')

  comentarios = xlrd.open_workbook('destinoResumen.xlsx')
  pagina_comentarios       = comentarios.sheet_by_index(0)

  ponderaciones_vendedores = {}

  comentarios_ponderados = []

  if os.path.exists(os.getcwd() + '\\vendedores_mejor_ponderados.txt'):
    os.remove('vendedores_mejor_ponderados.txt')

  archivo_vendedores = open('vendedores_mejor_ponderados.txt', 'a')

  for i in range(1, pagina_comentarios.nrows):

    pagina_comentario = pagina_comentarios.cell_value(rowx = i, colx = 0)
    vendedor = pagina_comentarios.cell_value(rowx = i, colx = 2)

    cant_comentarios = 0
    ponderacion_total = 0

    if (len(pagina_comentario) > 0):
      comentarios_pagina =  (pagina_comentarios.cell_value(rowx = i, colx =
5)).split('|')
      textos = [elem for elem in comentarios_pagina if len(elem) > 0]

      for texto in textos:
        if (len(texto) > 0) and (not texto.isspace()):
          comment = comentario(texto, vendedor, pagina_comentario)

          # FREELING
          texto_freeling = obtenerSalidaFreeling(texto, 'tagged', [])
          comment.agregar_texto_freeling(texto_freeling)

          # Incremento el contador de comentarios
          cant_comentarios +=1

          for linea in texto_freeling.split('\n'):

            tokens = re.split(r'[ \t\r]', linea)
            # Si no es una linea vacia.
            if ((len(tokens) > 0) and (not empty_elems(tokens))):
              palabra = tokens[0]
              lema = tokens[1]
              etiqueta = tokens[2]
              if not filtrar_palabra(etiqueta):
```

```
                # Verficamos que no sea una palabra vacia.
                if not lema in stop_words:
                  lema_codif = unicode(lema).encode('utf-8')
                  comment.agregar_palabras_comentario(lema_codif)
                  comment.imprimir_texto_freeling()
                  query = prolog.query('elementoSubjetivo('+ lema+',Y)')
                  if list(query):
                    query = prolog.query('elementoSubjetivo('+ lema +',Y)')
                    valor = (list(query))[0]["Y"]
                    comment.incrementar_ponderacion(valor)

                    ponderacion_total += valor

            else:
              print 'Tokens vacios o elementos vacios'
          # Se agrega el comentario a la lista de comentarios ponderados
          comentarios_ponderados.append(comment)
    ponderaciones_vendedores[vendedor] =
ponderacion_total/float(cant_comentarios)

  ordenados = sorted(ponderaciones_vendedores.items(),
key=operator.itemgetter(1), reverse=True)

  # Finalmente me quedo con los 10 primeros de esos vendedores ponderados
  rango_max = 10 if (len(ordenados) >=10) else len(ordenados)

  for elem in xrange (rango_max):
    archivo_vendedores.write('Vendedor: ' +
unicode(ordenados[elem][0]).encode('utf-8') + ' - Ponderacion: ' +
str(ordenados[elem][1]) + '\n')

  archivo_vendedores.close()
except (OSError, IOError):
       print 'Ha ocurrido un error.'
```

Bibliografía

Links de interés (visitados al 5 de diciembre de 2014):

- Wikipedia: http://es.wikipedia.org/wiki/WebCrawler
- Mercado Libre: http://www.mercadolibre.com.uy/
- OLX: http://www.olx.com.uy/
- Amazon: http://www.amazon.com/
- Ali Express: http://es.aliexpress.com/
- Herramientas de web crawling:
    - http://www.openwebspider.org/
    - http://scrapy.org/
    - http://sourceforge.net/projects/archive-crawler/
- PIP (Python Installer Packages): https://pip.pypa.io/en/latest/
- Analizadores de texto:
    - http://textalyser.net/index.php
    - http://nlp.lsi.upc.edu/freeling/
- Comprobador de expresiones regulares: http://www.regexr.com/