Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Inteligencia Artificial II
Enero-Marzo 2017

Clasificación de Sitios Web Según su Contenido

D. MARÍN, 10-10419. D. RODRIGUES, 10-10615. L. DÍAZ, 11-10293.

Resumen

Mediante algoritmos de machine learning se clasifican dominios web. Particularmente, para este caso se clasificaran los dominios que contengan información sobre Software relacionado con negocios y utilizando los algoritmos Naive Bayes, SVM y SGD. Para obtener el conjunto de datos utilizado, se seleccionó un conjunto de dominios relacionados con Software y negocios mediante el API de Alexa y se extrajo su contenido con el API de Common Crawl, el cual es procesado mediante la herramienta Weka para obtener una estructura manejable. Se aplicaron una serie de técnicas de minería de texto, se utilizaron diccionarios de stopwords para deshacernos de símbolos y palabras demasiado comunes que podrían generar ruido posteriormente en el estudio, se hizo stemming con la técnica Snowball (biblioteca y bibliotecario hacen match por poseer bibliotec) y se hizo la tokenización con una restricción mínima de frecuencia de 2, aplicando el método de Ngram donde escogimos frases de tamaño 1 a 3. Se calcularon las coordenadas de los vectores de cada instancia aplicando tf-idf.

Palabras claves: SVM, Naive Bayes, SGD, Software, Negocios, Minería de texto.

1 INTRODUCCIÓN

En 2013 habían registrados más de 250 millones de dominios, según "*The Domain Name Industry Brief*" emitido por la compañía americana Verisign. Se especula que actualmente existan más de 300 millones de dominios registrados. Esta enorme cantidad de dominios conlleva a un océano de información (y desinformación) que podría obstaculizar la búsqueda de algún tópico en específico. Este problema afecta a todos los usuarios de la red, desde los que están inmersos en campos relacionados con las ciencias hasta el usuario con menos experiencia, pero con acceso a la red.

De aquí surge la motivación de este proyecto. Debido a que el dominio del problema señalado es muy extenso, se diseñó una solución a un subconjunto del problema. Específicamente clasificar dominios que contengan información relacionada con Software y negocios del resto de los dominios. La razón por la que se decidió solucionar este problema en particular, es la necesidad que existe en el campo de desarrollo de software de obtener cierta información. La intención de separar los dominios relacionados con Software ó negocios es encontrar diferentes herramientas, como por ejemplo lenguajes, frameworks, soluciones SaaS (Software as a Service), librerías, etc. que puedan ayudar a solucionar algún problema específico de la mejor manera posible.

Para solucionar el problema planteado, primero se seleccionaron manualmente los dominios que contuvieran información relacionada con Software ó negocios, asumiendo la correctitud de la información en la API de Alexa. Luego se procedió a minar el contenido de las páginas fuentes de los dominios seleccionados en el paso anterior, específicamente considerando principalmente la información contenida en las etiquetas de: títulos, encabezados y párrafos. Posteriormente, esta información es filtrada para facilitar su procesamiento y permitir la elaboración del modelo usado en el paso final para lograr la separación exitosa de las clases de dominios especificadas anteriormente.

2 DETALLES DE IMPLEMENTACIÓN

Para buscar las categorizaciones se buscó una plataforma que brinda esta información. Alexa, una herramiento para SEO creada por Amazon que permite buscar sitios indexando por la categoría deseada. Se seleccionaron sitios en las categorías de computers/software/business y además, otras distintas con el motivo de balancear los datos. Luego se usó una herramienta denominada Common Crawl. Éste un movimiento open source que cuenta con terabytes de páginas web. A su vez, esta data está almacenada en un servicio en aws gratuito, por lo que puede ser accedida a través del protocolo http.

Una vez escogidos los sitios web y habiendo obtenido la información que estos contienen se procedió a construir un archivo csv que contuviera el dataset que luego sería transformado a .arff haciendo uso de la herramienta Weka, para posteriormente vectorizados a través con esta misma herramienta. Se hizo uso de técnicas como stemming, diccionarios de stopwords y se hizo una tokenización basada en NGram.

2.1 Conjunto de datos

El conjunto de datos de 119 instancias fue obtenido mediante el API de Alexa Internet, de donde se seleccionaron 60 sitios web de software y negocios de entre los más populares según los rankings de Alexa, así mismo se seleccionaron 59 sitios web pero esta vez relacionados con diferentes categorías como salud, economía, entretenimiento, etc. Para cada instancia se obtuvo su dominio, título principal y texto visible en el HTML del sitio web como atributos de la instancia, estos datos se procesaron haciendo uso de la librería BeautifulSoup de python.

2.2 Preprocesamiento de los datos

Dada una lista de los urls que compondrán las instancias del conjunto de datos, para cada una de ellas se obtiene el archivo HTML correspondiente al sitio web. Este archivo fue descargado del servicio web de Amazon haciendo uso del API de Common Crawl, que devuelve la información en un archivo comprimido para ahorrar espacio, la descarga y descompresión del archivo se realizaron mediante el algoritmo *download_page*, que devuelve toda la información contenida en el HTML de sitio web.

```
def download_page(record):
      offset, length = int(record['offset']), int(record['length'])
      offset_end = offset + length - 1

      prefix = 'https://commoncrawl.s3.amazonaws.com/'

      resp = requests.get(prefix + record['filename'],
            headers={'Range': 'bytes={}-{}'.format(offset, offset_end)})

      raw_data = StringIO.StringIO(resp.content)
      f = gzip.GzipFile(fileobj=raw_data)

      data = f.read()
      response = ""

      if len(data):
            try:
              warc, header, response = data.strip().split('\r\n\r\n', 2)
            except:
              pass

      return response
```

Para extraer el título principal y el contenido visible del HTML se utilizó el parser de HTML que provee BeutifulSoup. Para el título principal, con el algoritmo *extract_title* se encontraron todas las etiquetas *<title>* del documento y se seleccionó la primera de ellas, mientras que para el resto del texto visible se utilizó la función *extract_text* que parsea y devuelve todo el texto del documento HTML sin tomar en cuenta las etiquetas. Los datos fueron procesados para eliminar espacios innecesarios y saltos de línea.

```
def extract_title(html_content):
      parser = BeautifulSoup(html_content)
      title = parser.find_all("title")
      return title[0].contents[0].encode('ascii','ignore').strip("\n")


def extract_text(html_content):
      parser = BeautifulSoup(html_content)

      for script in parser(["script", "style"]):
      script.extract()   # rip it out

      text = parser.get_text()
      text = text.replace("\"", "\'")
      text = " ".join(text.split())
      return text
```

Finalmente los datos fueron agregados a un archivo .csv y catalogados con 1 si el dominio pertenece a la categoría de Software y Negocios y 0 de lo contrario.

2.4 Procesamiento de los datos.

Una vez introducido el archivo .csv a la herramienta weka se genera un archivo .arff con el cual trabajaremos de ahora en adelante. Debimos transformar el atributo de texto usando el filtro StringToWordVector, para reducir el universo usamos solo palabras con un mínimo de frecuencia igual a 2. Aplicamos el algoritmo para stemming llamado Snowball (biblioteca y bibliotecario hacen match por poseer bibliotec) y usamos un diccionario de stopwords en el que incluimos además de los términos comunes, todos los posibles símbolos, meses del año y días de la semana.

2.5 Solución

Claramente se está aplicando un aprendizaje supervisado, dado que se asume que la data está clasificada correctamente de antemano. Según la investigación realizada, Naive Bayes es una buena opción cuando se trata con text mining por lo que fue una de las opciones que escogidas, además de la que no podía faltar, SVM (Support Vector Machine) y finalmente un SGD (Stochastic Gradient Descent).

3 RESULTADOS

Se aplicaron dos métodos de validación, separando los datos de entrenamiento de los de prueba (70/30) y usando Cross Validation. Acorde a la investigación realizada Naive Bayes fue el clasificador que obtuvo los mejores resultados incluso contra SVM.

Usando cross validation:

	TP	FP	TN	FN	Recall
Naive Bayes	51	8	55	5	0,891 %
SVM	45	14	59	1	0,874 %
SGD	46	13	49	11	0,798 %

Usando train/test split (70%)

	TP	FP	TN	FN	Recall
Naive Bayes	15	1	19	1	0,944 %
SVM	14	2	20	0	0,944 %
SGD	12	4	16	4	0,778 %

4 CONCLUSIONES

El problema resuelto es una pequeña muestra del gran poder que proveen los algoritmos de aprendizaje de máquina. Prácticamente, todos los conjuntos relativamente grandes de datos (significativos), pueden ser procesados con un algoritmo de aprendizaje de máquina para obtener información que podría ser de interés para él que manipula los datos o algún tercero .También sirve para reconocer los pasos a seguir al momento de presentarse un problema de este tipo y la relevancia de diseñar la solución más acorde al problema. Al ser éste un problema de clasificación (Aprendizaje supervisado), existen diferentes alternativas para solucionarlo. En este proyecto, se utilizaron tres algoritmos diferentes para resolver el problema (individualmente):

- Naive Bayes
- SVM (Support Vector Machine)
- SVG (Stochastic Gradient Descent)

Los resultados indicaron que el algoritmo que resuelve el problema con mayor efectividad es Naive Bayes. Muy cerca se encuentra SVM y SVG es significativamente inferior.

5 REFERENCIAS

1. Ian H. Witten, Eibe F, Hall M. 2011. *Data Mining: Practical Machine Learning Tools and Techniques.* 3ª ed. Morgan Kaufmann Publishers. ISBN 978-0-12-374856-0
2. Pang-Ning T, Steinbach M, Kumar V. 2006. *Introduction to Data Mining*. [Consulta: Marzo 2016]. Disponible en: http://www-users.cs.umn.edu/~kumar/dmbook/index.php#item4
3. *Common Crawl: We build and maintain an open repository of web crawl data that can be accessed and analyzed by anyone.* [Consulta: Marzo 2016]. Disponible en: http://commoncrawl.org/.
4. *Weka 3: Data MIning Software in Java*. [Consulta: Marzo 2016]. Disponible en: http://www.cs.waikato.ac.nz/ml/weka/index.html
5. *Alexa: Find, Reach, and Convert Your Audience with Marketing That Works.* [Consulta: Marzo 2016]. Disponible en: http://www.alexa.com/