Clasificación de anuncios en Revolico

Ernesto Luis Estevanell Valladares, Rodrigo Sua García Eternod

Facultad de Matemática y Computación,
Universidad de la Habana, Habana, Cuba
{e.estevanell,r.garcia}@estudiantes.matcom.uh.cu

www.github.com/EEstevanell/Revolico-Category-Estimator

Resumen En este trabajo se propone una solución a la clasificación de anuncios de Revolico mediante el uso de técnicas de aprendizaje supervisado. Para resolver esta tarea se implementó un *crawler* para la extracción de corpus y poder realizar entrenamiento de los métodos. Entre las técnicas usadas se encuentran **Naive Bayes**, **Support Vector CLassifier** y **Regresión Logística**, las cuales han demostrado tener buenos resultados para la clasificación de texto.

Keywords: Clasificador, Machine Learning, Procesamiento de texto, Aprendizaje supervisado

1. Introducción

El procesamiento del lenguaje natural, abreviado PLN,es un campo de las ciencias de la computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano. El PLN se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio del lenguaje natural, es decir, de las lenguas del mundo...Los modelos aplicados se enfocan no solo a la comprensión del lenguaje en sí, sino a aspectos generales cognitivos humanos y a la organización de la memoria.

Una de las tareas más habituales del procesamiento natural del lenguaje es la clasificación de textos. El objetivo de esta es clasificar automáticamente en una o más categorías definidas previamente. Esto tiene mucha aplicación hoy en día tanto para el mundo científico como comercial. Por ejemplo las noticias del periódico se clasifican por temas, los usuarios de una tienda pueden ser clasificados en grupos acorde a su opinión sobre un producto o una marca. El objetivo de este trabajo es dado un anuncio, predecir la categoría en la que se ubica. Esto requiere la implementación de un *crawler* que permita recolectar los anuncios de Revolico.Además, el proyecto requiere implementar, evaluar, comparar y proponer algoritmos que permitan organizar y clasificar correctamente los nuevos anuncios.

2. Estado del Arte

Existen 3 formas principales de hacer la clasificación de texto:

1. **Sistemas basados en reglas**: Clasifican el texto utilizando una serie de reglas previamente definidas que detectan números, caracteres o términos específicos, lo cual también es conocido como expresiones regulares. Sin embargo este método generalmente es ineficiente ya que requiere un esfuerzo manual, da falsos positivos y no tiene bien en cuenta el contexto de las oraciones.
2. **Aprendizaje no supervisado**: Este tipo de algoritmos consisten en tratar de descifrar una estructura natural en los datos, por lo que buscan patrones y estructuras similares par agruparlos en *clusters*, lo cual permite la clasificación de los textos.
3. **Aprendizaje supervisado**: La clasificación supervisada de texto es hecha una vez se han definido las posibles categorías. El modelo se entrena con un corpus previamente clasificado y después se la pasa la información sin clasificar para obtener una de las salidas deseadas de forma automática.

Dentro de esta última forma de clasificación existen varios algoritmos típicamente usados.

2.1. Naive Bayes

Es una familia de algoritmos probabilísticos que se basan en aplicar el teorema de Bayes asumiendo la condición ïngenua"de la independecia condicional entre cada par de *features*.Este teorema calcula la probabilidad $P(c|x)$ donde c es la clase de las posibles salidas y x es la instancia a clasificar

$P(c|x) = P(x|c) * P(c)/P(x)$

Naive Bayes es muy usado en problemas de procesamiento natural del lenguaje. Para calcular la categoría correspondiente a un texto, calcula la probabilidad de pertenecer a cada una de las posibles clasificaciones y se queda con la que mayor dió.

2.2. Support Vector Classifier

SVC es un modelo utilizado para problemas de clasificación. Su objetivo es ajustar los datos del corpus brindado para devolver el hiperplano que mejor divide o categoriza nuestros datos. A partir de ese momento se le pueden introducir nuevos datos para que nos devuelva la clase correspondiente a cada uno.

2.3. Regresión Logística

La regresión logística es una técnica de aprendizaje que proviene del campo de la estadística. Con ella se mide ña relación entre la variable dependiente(afirmación que se desea predecir) y el conjunto de categorías disponibles a clasificar. Para ello se utiliza una función logística que determina la probabilidad de la variable dependiente Este es ampliamente usado debido a su simplicidad.

2.4. Aprendizaje profundo

Esta forma de aprendizaje consiste en un conjunto de algoritmos y técnicas basadas ene el funcionamiento del cerebro humano. La clasificación de texto es una de las aplicaciones que posee y ha demostrado tener resultados bastantes precisos, sin embargo requiere grandes cantidades de datos en comparación con otros algoritmos de *machine learning*. A diferencia de **SVM** o **Naive Bayes** que llega un momento en que se estancan de cierta forma al añadir más valores para entrenar, los algoritmos de d*deep learning* siguen mejorando. Las redes neuronales profundas, específicamente las **Redes Convolucionales** y las **Redes Recurrentes**, son las arquitecturas mas usadas para este tipo de tarea. [1,2,9,6,7,8,4,10]

3. Propuesta de Solución

Se propone una aplicación web que permite obtener, para un texto determinado, cuál sería su clasificación como anuncio en Revolico[1].

Dicha propuesta se compone de tres módulos principales:

- *Web app* (3.1)
- *Crawler* (3.2)
- *Engine* (3.3)

3.1. Web App

Se implementó un servicio Web utilizando **flask**. La interfaz de usuario propuesta permite la interacción del usuario con las herramientas presentes en los otros módulos.

Se ofrecen las siguientes funcionalidades:

- Iniciar un *crawler* para la recolección de anuncios de Revolico.
- Entrenar los modelos propuestos en *Engine*(3.3) con un corpus determinado.
- Obtener una clasificación para un texto determinado por cada estimador.

3.2. Crawler

Utilizando ***Scrapy***[2] se implementó una araña que permitiese la extracción de anuncios de Revolico con su correspondiente clasificación con el objetivo de construir un ***corpus***. Dicho ***dataset*** fue utilizado por el *Engine*(3.3) para entrenar el estimador implementado.

Se identificaron cinco categorías diferentes:

- Compra-Venta
- Empleo

[1] Popular sitio de compra y venta cubana
[2] *framework de python para crawlers*

- Computadoras
- Viviendas
- Autos

Se utilizó la versión *offline* de Revolico dadas las condiciones de conexión. Se pudieron observar algunas diferencias con el sitio real en cuanto a la estructura de las páginas pero, contando con la gran distribución de esta versión gracias al "paquete", se decidió seguir utilizándola.

Se implementó una política de ordenación **LIFO** y se estableció como **URL** semilla la raíz de Revolico. Una vez satisfecha la cantidad de anuncios de una categoría determinada, no se acceden a links de dicha categoría. De esta manera garantizamos una exploración balanceada por cada categoría, facilitando la recolección de anuncios.

3.3. Engine

Se implementó un *pipeline* de clasificación que permite variar sus componentes para lograr un rendimiento adecuado.

El *pipeline* propuesto consiste de las siguientes etapas:

- preprocesamiento
  1. tokenización
  2. *stemming*
  3. vectorización
- clasificación

3.4. Preprocesamiento

El preprocesamiento de los datos es una etapa fundamental de un *pipeline* de *machine learning*. Específicamente, al trabajar con texto, se necesitan realizar una serie de pasos para convertir los datos en una entrada válida para un método del dominio tratado.

Entre las componentes tratadas en nuestra propuesta se encuentran:

- tokenizador
- *stemmer*
- vectorizador

En la práctica, una selección incorrecta de un método en algún paso puede llevar a un rendimiento bajo del *pipeline* de clasificación. Además, existe una amplia gama de posibles técnicas para las funcionalidades anteriores, lo que hace de este un problema a lidiar.

Se presenta un estimador general que permite seleccionar, a modo de parámetro, un tokenizador o vectorizador a utilizar, dando facilidad para realizar un proceso de búsqueda sobre el espacio de decisiones. Se utilizó por defecto la función

word_tokenize[3] (**nltk**) como tokenizador mientras que ***CountVectorizer***[4] se seleccionó como vectorizador por defecto.

Se utilizó la popular clase de *stemming* ***SnowballStemmer*** de **nltk**.

3.5. Clasificación

Se propone el uso de tres clasificadores:

- Naive Bayes [5]
- Support Vector Machine (Linear kernel) [3]
- Logistic Regression

4. Comparando los clasificadores

Las pruebas se realizaron sobre dos datasets:

- Un dataset compuesto por 4000 anuncios con las 5 categorías contempladas en Revolico (3.2).
- El *dataset Movie Reviews* encontrado en ***nltk_data***.

Comparamos 3 *pipelines* distintos variando únicamente los clasificadores utilizados en cada caso (por simpleza nos referiremos a cada *pipeline* como un estimador). Cada estimador se compuso de la siguiente manera:

1. tokenizer: Función ***word_tokenize*** de **nltk**.
2. stemmer: ***SnowballStemmer*** de **nltk**.
3. vectorizer: ***CountVectorizer*** de **scikit_learn**.

Con cada estimador se realizaron pruebas en ambos *datasets*. Cada prueba consistía en obtener la precisión media mediante validación cruzada de *30-folds* como se muestra en la tabla 1. Las figuras 1, 2, 3 muestran las curvas de aprendizaje (mostrando precisión) de cada estimador con el dataset Revolico utilizando validación cruzada de *5-folds* por cada tamaño utilizado.

Dataset	**Naive Bayes**	**SVM (Linear kernel)**	**Logistic Regression**
Movie Reviews	80.75	83.34	**84.60**
Revolico	79.10	**85.44**	85.04

Tabla 1. Índice de precisión de clasificación medio de los estimadores sobre los *datasets Movie Review* y Revolico.

[3] Vectorizador recomendado por los creadores para el caso general.
[4] Vectorizador que consiste en obtener una matriz Término-Documento con los valores tf computados

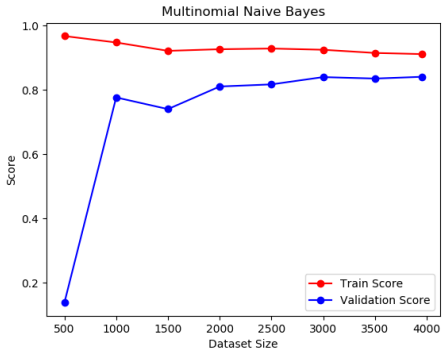

Figura 1. Curva de aprendizaje del estimador con Naive Bayes

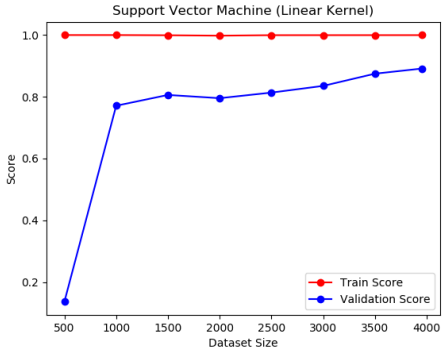

Figura 2. Curva de aprendizaje del estimador con SVM (Linear Kernel)

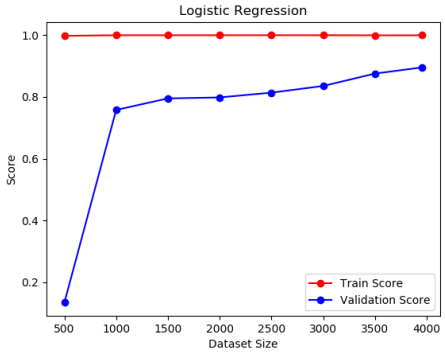

Figura 3. Curva de aprendizaje del estimador con Logistic Regression

5. Resultados

Se aprecian resultados similares entre los estimadores ***SVM*** y ***Logistic Regression*** para ambos *datasets*. Sin embargo, se nota una ligera desventaja de ***Naive Bayes*** ante sus contrincantes. Teniendo en cuenta que en el caso de Revolico existen 5 clasificaciones distintas, se han obtenido resultados muy satisfactorios con cada modelo.

Gracias a las curvas 1, 2, 3 podemos observar que los modelos presentan poco nivel de *bias* y un ligero nivel de *variance* que podría ser resuelto mediante la inclusión de más datos.

6. Conclusiones

Hemos logrado implementar 3 estimadores que alcanzan resultados muy satisfactorios en los *datasets* de prueba. Se construyó un crawler capáz de construir un corpus tageado a partir del sitio de anuncios clasificados Revolico. Además, se ofrece un servicio web que proporciona una interfaz de usuario intuitiva para la utilización de las herramientas implementadas.

Referencias

1. Text Classification. *monkeylearn.com*.
2. Shivam Bansal. A comprehensive guide to understand and implement text classification in python, Apr 2018.
3. Susan Dumais et al. Using svms for text categorization. *IEEE Intelligent Systems*, 13(4):21–23, 1998.
4. gk. Text classification using neural networks, Jan 2016.
5. Ashraf M Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence*, pages 488–499, 2004.
6. Kamran Kowsari. Text classification algorithms: A survey, May 2019.
7. Susan Li. Multi-class text classification model comparison and selection, Sep 2018.
8. Susan Li. Multi-class text classification with scikit-learn, Feb 2018.
9. Swayam Mittal. Deep learning techniques for text classification, Aug 2019.
10. Rushikesh Pupale. Support vector machines(svm) an overview, Jun 2018.