Procesamiento del Lenguaje Natural, Revista nº 57, septiembre de 2016, págs. 173-176 recibido 27-03-2016 revisado 11-04-2016 aceptado 25-04-2016

Sistema de predicción de peticiones de trabajos y servicios en sectores profesionales

Prediction system for job and service requests in professional sectors

Christian Moreno Bermúdez
Universidad de Jaén
Campus Las Lagunillas, s/n, 23071 Jaén
christianmorenobermudez@gmail.com

Arturo Montejo Ráez
Universidad de Jaén
Campus Las Lagunillas, s/n, 23071 Jaén
amontejo@ujaen.es

Resumen: El presente trabajo presenta un sistema que predice peticiones de trabajos y servicios en formato de texto en categorías o sectores profesionales. Se realiza una comparativa de distintos algoritmos de Categorización Automática de Textos para evaluarlos y construir el sistema. El sistema forma parte de una aplicación web que intermedia entre particulares que demandan presupuestos sobre trabajos y profesionales que buscan clientes y ofertan servicios.
Palabras clave: Categorización Automática de Textos, Minería de Textos, Minería de Datos, profesionales, presupuestos.

Abstract: System that predicts job requests and services in text format into categories or sectors. A comparison of different algorithms for Automatic Text Categorization is performed in order to build the final system. The system is part of a web application that mediates between individuals who demand estimates about jobs and professionals who seek clients and offer services.
Keywords: Automatic Text Categorization, Text Mining, Data Mining, professionals, budgets.

1 Introducción

En la actualidad, existe la necesidad de encontrar a profesionales que satisfagan las peticiones de personas en cuanto a trabajos y servicios se refiere, por ejemplo, para realizar una reforma en el hogar. Cuando una persona necesita de los servicios de un profesional debe consultar directorios de empresas, buscadores, preguntar a otras personas, etc. Si desea comparar presupuestos de varios profesionales, el proceso anterior se repite para cada profesional. Por otra parte, profesionales y empresas necesitan ofertar sus servicios a clientes para el desarrollo de su actividad.

El sistema de predicción, que se explicará a continuación, forma parte de una aplicación web desarrollada como prueba de concepto. Esta aplicación permite al particular registrar su necesidad mediante una descripción en formato de texto. Un ejemplo sería: *"Quisiera presupuesto para instalar rejas en 4 ventanas, las dimensiones son de 1,5x1,5m"*. Una vez enviada la petición, el sistema de predicción la clasifica en el sector profesional correspondiente, que en el ejemplo anterior coincidiría con "Carpintería Metálica". Posteriormente, la aplicación se encarga de avisar a los profesionales correspondientes dados de alta en dicho sector para que manden sus presupuestos. Este sistema constituye, pues, una demostración práctica del uso de la categorización automática de textos.

Este proyecto está parcialmente financiado por el Gobierno de España a través del proyecto REDES (TIN2015-65136-C2-1-R).

Para realizar el sistema se llevan a cabo las siguientes etapas: recopilación y preparación de datos que se explica en la sección 3, selección de algoritmos a comparar en la sección 4 y evaluación de resultados en la sección 5.

2 Antecedentes

La Categorización y clasificación Automática de Textos (Sebastiani, 2002) es una de las tecnologías del lenguaje humano que mayor aplicación ha demostrado (Jackson y Moulinier, 2007). Entendemos la categorización automática de textos como la asignación de un texto a una o varias categorías (mono-etiqueta o multi-etiqueta). La

gran mayoría de los algoritmos orientados a la clasificación, entre los que destacan las *Support Vector Machines* (SVM) (Joachims, 1998), suelen ser de tipo binario, esto es, el algoritmo clasifica el texto a una de dos posible categorías. Sobre estos clasificadores binarios se construyen los multiclase (una de varias categorías) y los multietiqueta (una o más de varias categorías posibles). En cualquier caso, actualmente se ha encontrado niveles de precisión altos que posibilitan el uso de estos algoritmos con garantías en aplicaciones reales, como el filtrado de información en la web (Gentili et al., 2001), procesamiento de textos legales (Cardie et al., 2008) o la clasificación de artículos para la Física de Altas Energías (Montejo-Ráez et al., 2005).

3 Recopilación y preparación de datos

Es necesario un conjunto de peticiones de trabajos clasificados para entrenar el sistema de predicción. En Internet encontramos diferentes sitios web que ayudan a resolver la problemática planteada en la introducción y muestran un amplio conjunto de ejemplos de peticiones reales ya categorizadas. Nosotros usaremos la herramienta *wget*[1] de UNIX para obtener un total de 43.601 documentos HTML, donde un documento contiene una petición de trabajo clasificada en una de 24 categorías posibles.

Se eliminan aquellas categorías que son una agregación de varias categorías, por ejemplo, "Reformas" que incluye ejemplos donde necesitan fontaneros, carpinteros, albañiles, etc. simultáneamente. A pesar de ser ejemplos válidos, pertenecen a un problema de clasificación multi-etiqueta en el que no se centra este proyecto.

Tras el sesgo anterior, el conjunto de dato queda con 31.056 ejemplos y 16 categorías. Se genera una muestra aleatoria de 10.000 ejemplos para realizar las pruebas.

A continuación, se aplican los procesos que construyen el índice de un buscador sobre ficheros de texto:

1. Eliminación de caracteres indeseables.
2. Eliminación de palabras que carecen de significado o *stopwords*.
3. Extracción de raíces, con la ayuda de un *stemmer*[2] para el español.
4. Aplicación del Modelo Espacio Vectorial, para generar la matriz de pesos que dará soporte al clasificador. Se obtiene una matriz con 7.652 atributos: *calder, repar, papel, termostat*, etc.

Para mejorar el tiempo de entrenamiento, se reduce el tamaño de la matriz eliminando aquellos términos que aparecen menos de 10 veces en todo el conjunto de datos. El conjunto de atributos desciende a 1.283.

Por último, se plasman los datos en formato *.arff*. Este es el formato usado por la herramienta Weka[3], que contiene los diferentes algoritmos de minería de datos que usaremos para la comparativa.

Antes de lanzar los experimentos, conviene reducir aún más la dimensión del conjunto de datos aplicando un método de selección de atributos, como el filtro basado en consistencia (Liu y Setiono, 1996), que mejora en gran medida el tiempo de entrenamiento e incluso la tasa de acierto entre 1 y 3 puntos porcentuales.

Finalmente, el conjunto resultante contiene:

- Nº de instancias: 9.984
- Nº de atributos: 280
- Nº de categorías: 16

4 Selección de algoritmos

Para llevar a cabo el experimento, se ha creído conveniente elegir cinco familias de algoritmos diferentes en las que se incluyen los siguientes algoritmos y parámetros:

- Algoritmos basados en probabilidad:
  - **BayesNe**t (O. Pourret et al., 2008), redes bayesianas con los parámetros por defecto definidos en Weka.
  - **Naïve Bayes** (Rish, 2001), con los parámetros por defecto.
- Máquinas de Vectores de Soporte:
  - **LibSVM** (Chih-Chung y Chih-Jen, 2013), combinando:
    - kernel: linea, polinómico, de base radial y sigmoidal.
    - C (factor de complejidad): 0,125, 1 y 2.
  - **SMO** (Platt, 1998), combinando:
    - kernel: polinómico, gaussiano.
    - C (factor de complejidad): 0,125, 1 y 2.

[1] https://www.gnu.org/software/wget/

[2] http://snowball.tartarus.org/

[3] http://www.cs.waikato.ac.nz/ml/weka/

    - Exponente (solo para kernel polinómico): 0,125, 1 y 2.
    - Gamma (solo para kernel gaussiano): 0,1, 0,01 y 0,001.
- Algoritmos basados en proximidad:
    - **KNN** (Altman, 1992), con los siguientes número de vecinos (k): 1, 3, 5 y 7.
- Algoritmos basados en reglas:
    - **RIPPER** (William, 1995) o JRip, con los parámetros por defecto.
- Algoritmos basados en árboles:
    - **C4.5** (Quinlan, 1993) o J48, con los parámetros por defecto.
    - **RandomForest (**Breiman, 2001), con número máximo de árboles: 20, 50, 80 y 110.

Esta combinación de algoritmos y parámetros da como resultado un total de 42 clasificadores entrenados en Weka.

Aquellos que mejor se comporten, constituirán las bases para construir 3 meta-clasificadores diferentes, con los que se espera una mejora en los resultados:

- **StackingC** (Seewald, 2002): Se seleccionarán los cuatro mejores algoritmos anteriores para construir un clasificador por regresión.
- **Bagging** (Breiman, 1996): Bagging escogiendo porciones del dataset del 33,33% con el mejor algoritmo base.
- **AdaBoostM1** (Freund y Schapire, 1996): Boosting con el mejor algoritmo base.

5 Evaluación de resultados

A continuación, se expone la Tabla 1 que incluye la comparativa de la tasa de acierto en predicción de los cuatro mejores algoritmos base:

ALGORIT	PARAM	ACIERTO
BayesNet	Por def.	81,32 %
LibSVM	C = 2, Kernel = Lineal	83,95 %
SMO	C = 2 , exp = 1, Kernel = Polin.	83,89 %
Random Forest	Tam = 80	81,53 %

Tabla 1: Comparativa de los mejores algoritmos base

Dado los resultados de la tabla anterior, se escoge el algoritmo LibSVM con factor de complejidad igual a 2 y kernel lineal para construir un meta-clasificador por Bagging y por Boosting. Para construir el meta-clasificador por Stacking se elige a los cuatro algoritmos de la tabla.

En este caso, se tiene en cuenta, además de la tasa de acierto, las métricas de Error Medio Absoluto y coeficiente de Kappa (Cohen, 1960). El Error Medio Absoluto verifica con qué grado de exactitud el sistema acierta, es decir, cómo de lejos están las predicciones de los datos reales. El coeficiente Kappa es una medida que muestra como de bueno es un clasificador estudiando la concordancia de los resultados obtenidos por varios clasificadores del mismo tipo. Valores cercano a 1 afirman una concordancia buena, mientras que valores cercanos a 0 muestran una concordancia debida exclusivamente al azar.

Tras el lanzamiento de las pruebas, se obtienen los siguientes resultados (véase Tabla 2):

ALGORIT	ACIERTO	E.M.A	KAPPA
StackingC	84,93 %	0,0307	0,8326
Bagging	82,90 %	0,0235	0,8094
AdaBoostM1	83,21 %	0,0470	0,8135

Tabla 2: Comparativa de meta-clasificadores

De los resultados que se muestran en la Tabla 2, se desprende que el algoritmo más prometedor para implementar el sistema de predicción es **StackingC**, ya que presenta la mayor tasa de acierto, una media del error absoluto en un intervalo razonable y el mejor estadístico de Kappa.

6 Implementación en la aplicación web

Para implementar el algoritmo en la aplicación web, se ha utilizado la API de Weka en lenguaje JAVA. En un proyecto a parte, se puede cargar el dataset de instancias y atributos con el que entrenar un objeto de la clase weka.classifiers.meta.StackingC. Posteriormente, el objeto se serializa y queda listo para recuperarlo y usarlo desde la aplicación web.

La secuencia de pasos a realizar en la aplicación es bastante simple. El usuario introduce la descripción de aquello que necesita en un formulario, junto a sus datos de

contacto. Seguidamente, el sistema de predicción clasifica la petición en el sector profesional que corresponde. Acto seguido, se notifica a los profesionales de dicho sector y próximos al usuario de que tienen una solicitud de presupuesto interesante. Los profesionales pueden contactar con el cliente y mandar sus presupuestos para que el usuario compare fácilmente y decida. Dado que el sistema no es infalible, el usuario puede modificar la categorización realizada manualmente.

7 Conclusiones y trabajo futuro

Hemos construido un sistema funcional para el enrutado de peticiones de trabajo a profesionales gracias a la aplicación de algoritmos de categorización de textos. Hemos evaluado su precisión con resultados que llevan a concluir la factibilidad de estas tecnologías en este dominio de cara a su explotación real. Cabe destacar que el sistema se trata de un prototipo en un contexto limitado. Es necesario un conjunto de ejemplos más rico tanto en descripciones como en categorías empresariales.

Por otra parte, sería muy interesante desarrollar un clasificador multi-etiqueta, ya que en muchos casos encontraremos descripciones que hacen referencia a varios sectores profesionales. Por ejemplo, en una descripción como *"quisiera mudarme a un piso nuevo y se necesitan trasladar todos los muebles. Además es necesario pintar el piso"* se debería avisar a tanto a profesionales de la mudanza como a pintores.

Bibliografía

Altman, N. S. 1992. *An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46 (3)*: 175–185.

Breiman, Leo. (1996). *Bagging predictors. Machine Learning.* 24(2):123-140.

Breiman, Leo. (2001). *Random Forests. Machine Learning* 45 (1): 5–32.

Cardie, C., Farina, C. R., Rawding, M., & Aijaz, A. (2008). *An eRulemaking Corpus: Identifying Substantive Issues in Public Comments.*

Chih-Chung Chang and Chih-Jen Lin (2013). *LIBSVM: A Library for Support Vector Machines.* National Taiwan University.

Cohen J. *A coefficient of agreement for nominal scales*. Educ Psychol Meas 1960; 20: 37-46.

Freund Yoav and Schapire Robert E. (1996): *Experiments with a new boosting algorithm.* 148-156.

Gentili, G. L., Marinilli, M., Micarelli, A., & Sciarrone, F. (2001). *Text categorization in an intelligent agent for filtering information on the Web.* 15(03), 527-549.

H. Liu, R. Setiono (1996): A *probabilistic approach to feature selection - A filter solution*. 319-327.

Jackson, P., & Moulinier, I. (2007). *Natural language processing for online applications: Text retrieval, extraction and categorization* .

Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features* (pp. 137-142). Springer Berlin Heidelberg.

Montejo-Ráez, A., Urena-Lopez, L., & Steinberger, R. (2005). *Text categorization using bibliographic records: beyond document content.* 119-126.

O. Pourret, P. Naim and B. Marcot. 2008. Bayesian Networks: *A Practical Guide to Applications*. Chichester, UK: Wiley.

Platt, John. 1998, *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.*

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning.*

Rish, Irina. 2001. *An empirical study of the naive Bayes classifier*.

Sebastiani, F. (2002). *Machine learning in automated text categorization. ACM computing surveys (CSUR), 34*(1), 1-47.

Seewald, A. K. (2002). *How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness*. 554-561.

William W. Cohen. 1995. *Fast Effective Rule Induction. In: Twelfth International Conference on Machine Learning*, 115-123.