Facultad de Ciencias, UNAM

Sistema de clasificación automática de documentos

Reconocimiento de Patrones y Aprendizaje Automatizado

Oscar Andrés Rosas Hernandez

15 de marzo de 2020

Índice general

Parte I

Marco Teórico

Capítulo 1

Aprendizaje Supervisado

Definimos al machine Learning como el área que estudia como hacer que las computadoras puedan aprender de manera automática usando experiencias (data) del pasado para predecir el futuro.

1.1. Definición

La mayoría del aprendizaje automático práctico utiliza aprendizaje supervisado. El aprendizaje supervisado es donde se tiene variable(s) de entrada (x) y una variable de salida (Y) y se utiliza un algoritmo para "aprender" la función que mapea la entrada a la salida.

$$Y = f(X) \tag{1.1}$$

El objetivo es aproximar la función tan bien que cuando tenga nuevos datos de entrada (x) pueda predecir las variables de salida (Y) para esos datos.

Se llama aprendizaje supervisado porque el proceso de un algoritmo que aprende del conjunto de datos de capacitación puede verse como un profesor que supervisa el proceso de aprendizaje.

Conocemos las respuestas correctas, el algoritmo realiza predicciones de forma iterativa sobre los datos de entrenamiento y es corregido por el profesor.

Los problemas de aprendizaje supervisados pueden agruparse en problemas de regresión y clasificación.

- Clasificación: Un problema de clasificación es cuando la variable de salida es una categoría, como $rojo$ o $azul$ o $enfermedad$ y $sin_enfermedad$.
- Regresión: Un problema de regresión es cuando la variable de salida es un valor contiuo, como $dolares$ o $peso$ o $probabilidad$.

Durante el entrenamiento, un algoritmo de clasificación recibirá una serie de datos con una categoría asignada. El trabajo de un algoritmo de clasificación es tomar un valor de entrada y asignarle una clase o categoría que se ajuste según los datos de training proporcionados. [1]

1.1.1. Accuracy

Esta dada por:

" De todos nuestros datos, que tanto % clasificamos correctamente. "

En otra palabras:

$$accuracy := \frac{TrueNegative + TruePositive}{All}$$

Ahora, "accuracy" no es siempre la mejor métrica, sobretodo cuando nuestros datos no estan balanceados, es decir cuando en cada una de nuestras clases tenemos una cantidad diferente de datos.

Capítulo 2

Arboles de Decision

Los árbol de decisión son algoritmos de machine learning que dividen progresivamente el conjuntos de datos en grupos de datos más pequeños en función de una característica descriptiva, hasta que alcanzan conjuntos que son lo suficientemente pequeños como para ser descritos por alguna etiqueta.

Requieren que tenga datos etiquetados, por lo que intentan etiquetar los nuevos datos en función de ese conocimiento. Estos son perfectos para resolver la clasificación.

La importancia de los árbol de decisión se basa en el hecho de que tienen muchas aplicaciones en el mundo real. Al ser uno de los algoritmos más utilizados, se aplican en varias industrias, por ejemplo:

- Los árbol de decisión se están utilizando en la industria de la salud para mejorar la detección de casos positivos en la detección temprana del deterioro cognitivo y también para identificar los principales factores de riesgo de desarrollar algún tipo de demencia en el futuro.
- Sophia, el robot que se hizo ciudadana de Arabia Saudita, utiliza algoritmos árbol de decisión para chatear con humanos. De hecho, los chatbots que usan estos algoritmos ya están brindando beneficios en industrias como los seguros de salud al recopilar datos de los clientes mediante la aplicación de encuestas innovadoras y chats amigables. Google adquirió recientemente Onward, una compañía que usa árbol de decisión para desarrollar chatbots que son excepcionalmente funcionales para brindar atención al cliente de clase mundial, y Amazon está invirtiendo en la misma dirección para guiar a los clientes rápidamente hacia un camino de resolución.
- Es posible predecir las causas más probables de las perturbaciones forestales, como los incendios forestales, la tala de plantaciones de árboles, la agricultura a gran o pequeña escala y la urbanización mediante la capacitación de árbol de

decisión para reconocer las diferentes causas de pérdida de bosques a partir de imágenes satelitales.

- Los árbol de decisión son excelentes herramientas para realizar análisis de sentimientos de textos e identificar las emociones detrás de ellos. El análisis de sentimientos es una técnica poderosa que puede ayudar a las organizaciones a aprender sobre las elecciones de los clientes y sus decisiones.
- Los árbol de decisión también se utilizan para mejorar la detección de fraudes financieros. El MIT demostró que podría mejorar significativamente el rendimiento de modelos alternativos de macine learning mediante el uso de árbol de decisión que fueron entrenados con varias fuentes de datos en bruto para encontrar patrones de transacciones y tarjetas de crédito que coincidan con los casos de fraude.

Los árbol de decisión son extremadamente populares por una variedad de razones, siendo su interpretabilidad probablemente su ventaja más importante.

Se pueden entrenar muy rápido y son fáciles de entender, lo que abre sus posibilidades a fronteras más allá de los muros científicos. Hoy en día, los árbol de decisión son muy populares en entornos empresariales y su uso también se está expandiendo a áreas civiles, donde algunas aplicaciones generan grandes preocupaciones. [2]

Parte II

Clasificación de Textos

Capítulo 3

El problema

El problema elegido fue el predecir la urgencia de un mensaje dado en contenido del mismo, siendo mas especificos intente clasificar la urgencia de un "support ticket".

El término describe la interacción entre un cliente y un representante de servicio. Es el elemento básico de cualquier trabajo relacionado con la experiencia del cliente: le permite a la empresa crear, actualizar y, con suerte, resolver cualquier problema que puedan tener sus usuarios finales.

3.0.1. Importancia de resolverlo

La técnica de los "support ticket" es muy usada en la industria, sobretodo en el área de ventas con el cliente y servicio técnico y suele ser que la cantidad de estos mensajes superan con creces a la cantidad de personas que pueden atenderlos por lo que se vuelve de vital importancia crear sistemas que nos permitan clasificar estos para priorizar de acuerdo a como se crear necesario, mi objetivo seria crear un sistema que pueda ayudar con la clasificación de estos tickets enfocados en su urgencia para que el personal pueda atenderlos usando una cola de prioridad.

Capítulo 4

El dataset / Business Understanding

Para poder solucionar este problema buscamos un dataset que fuera apropiedo en Kaggle, llegando a este: `https://github.com/karolzak/support-tickets-classification#22-dataset` / `https://www.kaggle.com/aniketg11/supportticketsclassification`.

Este fue creado como una demo que muestra cómo crear un modelo para análisis y clasificación de texto e implementarlo como un servicio web en la nube de Azure para clasificar automáticamente los tickets de soporte.

Este proyecto es una prueba de concepto realizada por Microsoft (equipo de Ingeniería de Software Comercial) en colaboración con Endava.

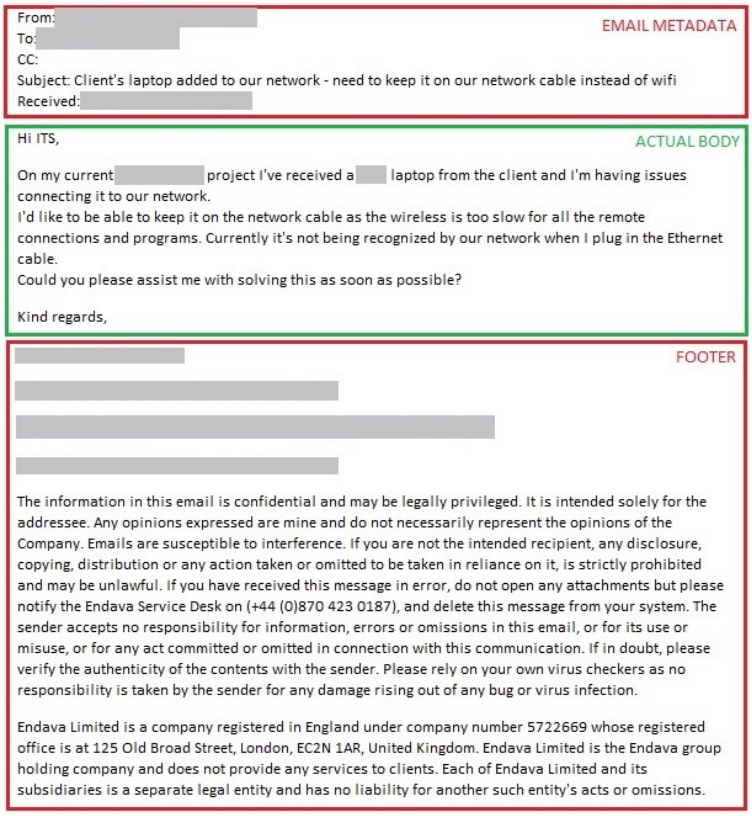
El conjunto de datos que fue utilizado son datos internos de Endavas importados de su sistema de servicio de asistencia.

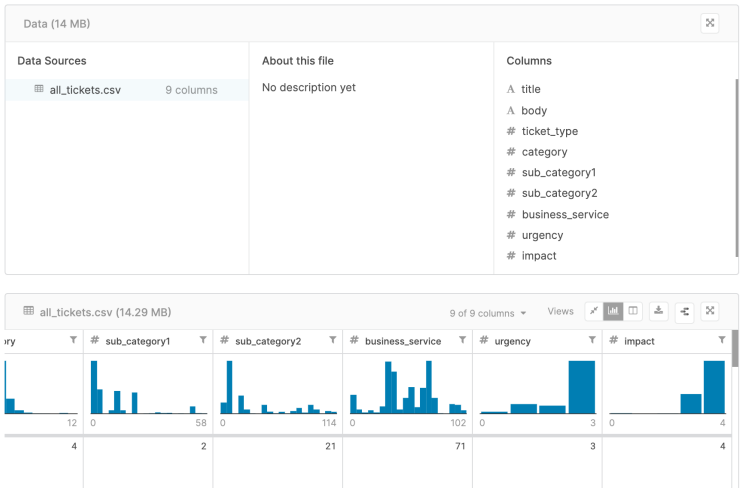
Pudieron recolectar alrededor de 50 mil tickets de asistencia clasificados con mensajes originales de los usuarios y etiquetas ya asignadas.

Ademas es importante notar que el dataset usado esta ya anonimizado y le fue quitada toda la información sensible.

Podemos intentar ver como esta conformado este dataset, siendo para nosotros importantes los campos de *body* y *urgency*.

Ademas es importante notar que el campo de urgencia tiene solo 4 niveles, siendo por ejemplo el nivel 3 considera “low”.


From:
To:
CC:
Subject: Client's laptop added to our network - need to keep it on our network cable instead of wifi
Received:
EMAIL METADATA
Hi ITS,
On my current project I've received a laptop from the client and I'm having issues connecting it to our network.
I'd like to be able to keep it on the network cable as the wireless is too slow for all the remote connections and programs. Currently it's not being recognized by our network when I plug in the Ethernet cable.
Could you please assist me with solving this as soon as possible?
Kind regards,
ACTUAL BODY
FOOTER
The information in this email is confidential and may be legally privileged. It is intended solely for the addressee. Any opinions expressed are mine and do not necessarily represent the opinions of the Company. Emails are susceptible to interference. If you are not the intended recipient, any disclosure, copying, distribution or any action taken or omitted to be taken in reliance on it, is strictly prohibited and may be unlawful. If you have received this message in error, do not open any attachments but please notify the Endava Service Desk on (+44 (0)870 423 0187), and delete this message from your system. The sender accepts no responsibility for information, errors or omissions in this email, or for its use or misuse, or for any act committed or omitted in connection with this communication. If in doubt, please verify the authenticity of the contents with the sender. Please rely on your own virus checkers as no responsibility is taken by the sender for any damage rising out of any bug or virus infection.
Endava Limited is a company registered in England under company number 5722669 whose registered office is at 125 Old Broad Street, London, EC2N 1AR, United Kingdom. Endava Limited is the Endava group holding company and does not provide any services to clients. Each of Endava Limited and its subsidiaries is a separate legal entity and has no liability for another such entity's acts or omissions.


Data (14 MB)
Data Sources
all_tickets.csv 9 columns
About this file
No description yet
Columns
title
body
ticket_type
category
sub_category1
sub_category2
business_service
urgency
impact
all_tickets.csv (14.29 MB)
9 of 9 columns
Views
sub_category1
sub_category2
business_service
urgency
impact

Capítulo 5

La propuesta

Mi hipotesis sería que usando este dataset y mediante aprendizaje supervisado (árboles de decisión) se podría crear un sistema que fuera capaz de clasificar con un gran alto nivel de exactitud (accuracy) (alrededor de un 85 %).

Los árboles de decisión fueron elegidos por los puntos anteriormente dichos, sobretodo por la gran capacidad que interpretabilidad y que es conocido que los árboles de decisión tienen un gran rendimiento en el analisis de textos.

En especial y debido a lo visto en clase se usara el J48 como clasificador. Tambien hablando mas detalladamente de el espacio de hipotesis tenemos que:

- Realiza aprendizaje por lote, que procesa todo el entrenamiento a la vez en lugar de aprendizaje incremental que actualiza una hipótesis después de cada ejemplo.
- Encuentra una sola hipótesis discreta.

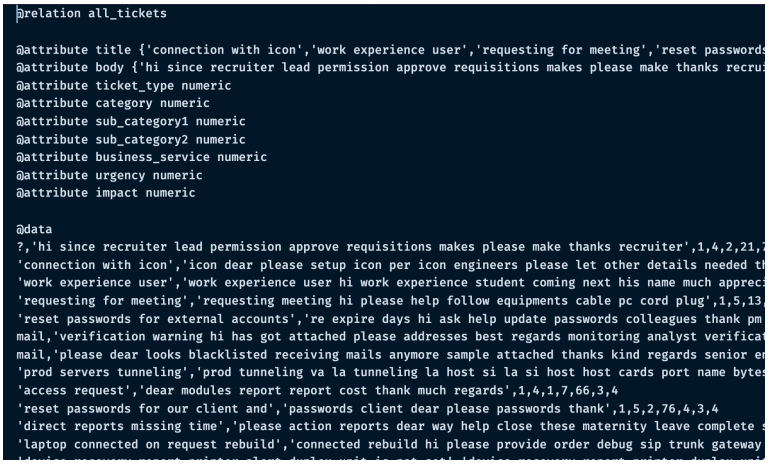
La principal desventaja de esta propuesta es que se va a usar stringWordToVector, lo que perderemos una gran cantidad de información, sobretodo del contexto, pues nos quedaremos con un vector de frecuencias (o de existencia) pero no hay forma de saber donde estaban esas palabras originalmente en el texto, esta información para este problema puede llevar a que el sistema no se desempeñe tan bien como debería.

Capítulo 6

La implementación

6.0.1. Preparación de los datos

Lo primero que tuvimos que hacer fue preparar los datos, el primer paso fue descargar el csv (all_tickets.csv) y abrirlo con weka para crear un archivo arff que pudieramos usar desde Java, ahora si, con ese archivo listo (all_tickets.arff).



Ahora, lo siguiente fue quedarnos solo con los campos que necesitamos, en este caso el body y la etiqueta, por lo que eliminamos todos los demas.

Ahora, tuvimos que usar dos filtros de weka para podemos poner los datos con los tipos apropiados. Primeramente tuvimos que usar el $NumericToNominal$ para poder poner las etiquetas como un dato nominal y no numerico.

después tuvimos que hacer algo similar con el body que por defecto weka lo tomaba como un dato nominal.

Ahora con ambos datos del tipo correcto ya podiamos usar $StringToWordVector$, a este tuvimos que darles varios parametros especiales (con la ayuda de $splitOptions$),

en especial buscamos reducir la cantidad de palabras a conservar, pasando de 1000 a 100, decidimos dejar la diferencia entre mayusculas y minisculas y tambien no usar un steamer pues no existe una compatibilidad total (ademas de que diferentes experimentos demostraron que su uso no afectaba significativamente el desempeño del ente), con est filtro aplicado, ahora si estabamos listo para clasificar, teniendo alrededor de 101 atributos.

6.0.2. Creación del clasificador

Esto fue mucho mas sencillo e igualmente buscamos crear el modelo mas sencillo que fuera capaz de resolver el problema, llegando a que el parametro que mas afectba el desempeño del sistema era la cantidad de elementos necesario para crear una hoja, un valor de 15 resulto ser mucho mas efectivo en reducir el tamaño del arbol y aumentar el desempeño.

Ahora tambien fue importante cambiar el tamaño del bache a 64, siendo una creencia común que conocía y comprobé que usar una potencia de 2 aumenta significativamente el tiempo de entrenamiento.

6.0.3. Evaluación

Lo que hicimos para evaluar al sistema fue partir nuestros datos, elegimos un 80 %, algo muy importante (y que me tomo dos horas darme cuenta y cambio el resultado del sistema de un 40 % a un 85 %) es que antes de partir los datos lso "revolvimos", para evitar que cierto grupo de etiquetas quedara sobrerepresentado en algun conjunto.

6.0.4. interpretación

Con esto listo creamos un evaluador y medimos el resultado de nuestro clasificador.

```
@attribute body string
@attribute urgency {0,1,2,3}
48549
101

Correctly Classified Instances        8295               85.4274 %
Incorrectly Classified Instances      1415               14.5726 %
Kappa statistic                          0.6738
Mean absolute error                      0.0867
Root mean squared error                  0.218
Relative absolute error                 37.9172 %
Root relative squared error             64.5128 %
Total Number of Instances             9710
```

En este podemos ver inmediatamente que nuestro sistema cumplio nuestro objetivo, logran un mas de 85 % de accuracy en el conjunto de prueba.

Ademas viendo a la matriz de confusión vemos que los errores se distribuyen de manera informe. Finalmente y como una posible forma de llevarlo a las manos de los usuarios cree un programa que usa los modelos y al que tu le pasas un texto por la terminal y te regresa su nivel estimado de urgencia.

```
→  Code git:(master) ✗ javac -cp weka.jar Tests.java && java -cp weka.jar:. Tests hello man
3
→  Code git:(master) ✗
```

6.0.5. Codigo

Codigo principal

Compilarse usando: javac -cp weka.jar Project.java && java -cp weka.jar:. Project y java v13

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.trees.J48;

import weka.core.Instances;
import weka.core.Utils;
import weka.core.tokenizers.WordTokenizer;
import weka.core.SerializationHelper;

import weka.filters.Filter;
import weka.filters.unsupervised.attribute.NominalToString;
import weka.filters.unsupervised.attribute.NumericToNominal;
import weka.filters.unsupervised.attribute.StringToWordVector;

class Project {
  /*Show all attribute names and type to the screen*/
  static void printAttibutes(final Instances dataset) {
    final var top = dataset.numAttributes();
    for (int i = 0; i < top; ++i)
      System.out.println(dataset.attribute(i));
  }

  public static void main(String[] args) throws Exception {
    final String DATASET_FILE = "all_tickets.arff";
    final double PERCENTAGE_SPLIT = 0.8;
    final int[] ROWS_TO_DELETE = new int[] {9, 7, 6, 5, 4, 3, 1};
    final String STWV_OPTIONS = "-R 1 -W 100 -N 0 -M 1";
    final String CLASSIFIER_OPTIONS = "-U -B -M 15 -batch-size 64";

    final String MODEL_SAVE = "j48.model";
    final String FILTER_SAVE = "stwv.filter";

    // Get the original dataset
    final var path = DATASET_FILE;
    var dataset = new Instances(new BufferedReader(new FileReader(path)));

    // Delete all the fields we will not use now
    for (final var index : ROWS_TO_DELETE)
      dataset.deleteAttributeAt(index - 1);

    // Cleaning / Data preparation
    // Transform urgency into nominal
    final var numericToNominal = new NumericToNominal();
    numericToNominal.setOptions(Utils.splitOptions("-R last"));
    numericToNominal.setInputFormat(dataset);
    final var datasetV2 = Filter.useFilter(dataset, numericToNominal);

    // Transform body into string
    final var nominalToString = new NominalToString();
    nominalToString.setOptions(Utils.splitOptions("-C first"));
    nominalToString.setInputFormat(datasetV2);
    final var datasetV3 = Filter.useFilter(datasetV2, nominalToString);
```

```
    printAttibutes(datasetV3);
    System.out.println(datasetV3.numInstances());

    // Transform body into a word vector
    final var stringToWordVector = new StringToWordVector();
    stringToWordVector.setOptions(Utils.splitOptions(STWV_OPTIONS));
    stringToWordVector.setInputFormat(datasetV3);
    final var datasetV4 = Filter.useFilter(datasetV3, stringToWordVector);

    dataset = datasetV4;
    System.out.println(dataset.numAttributes());

    // Create the classifier
    final var classifier = new J48();
    classifier.setOptions(Utils.splitOptions(CLASSIFIER_OPTIONS));
    dataset.setClassIndex(0);
    dataset.randomize(new java.util.Random(0));

    // Split and create a train and test dataset
    final var n = dataset.numInstances();

    final var percent = PERCENTAGE_SPLIT;
    final var trainSize = (int)Math.round(n * percent);
    final var testSize = n - trainSize;

    final var train = new Instances(dataset, 0, trainSize);
    final var test = new Instances(dataset, trainSize, testSize);

    // Build the classifier
    classifier.buildClassifier(train);

    // Evaluate the classifier using test dataset
    final var eval = new Evaluation(train);
    eval.evaluateModel(classifier, test);
    System.out.println(eval.toSummaryString());

    // Save them
    SerializationHelper.write(MODEL_SAVE, classifier);
    SerializationHelper.write(FILTER_SAVE, stringToWordVector);
  }
}
```

Codigo de pruebas del usuario final

```
@attribute body string
@attribute urgency {0,1,2,3}
48549
101

Correctly Classified Instances        8295               85.4274 %
Incorrectly Classified Instances      1415               14.5726 %
Kappa statistic                          0.6738
Mean absolute error                      0.0867
Root mean squared error                  0.218
Relative absolute error                 37.9172 %
Root relative squared error             64.5128 %
Total Number of Instances             9710
```

```
import java.io.BufferedReader;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintStream;
import java.util.Arrays;
import weka.classifiers.Classifier;
import weka.classifiers.trees.J48;
import weka.core.Instances;
import weka.core.SerializationHelper;
import weka.core.Utils;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.StringToWordVector;
class Tests {

  static void saveToFile(String filename, String text) throws Exception {
    try (final var out = new PrintStream(new FileOutputStream(filename))) {
      out.print(text);
    }
```

```
  }

  public static void main(String[] args) throws Exception {
    final var path = "test.arff";

    // Create necessary file using the data given
    saveToFile(path, String.format("@relation test\n"
                                       + "@attribute body string\n"
                                       + "@attribute urgency {0, 1, 2, 3}\n"
                                       + "@data\n"
                                       + "'%s',0",
                                   String.join(" ", args)));

    // Create the instance
    var tests = new Instances(new BufferedReader(new FileReader(path)));

    // Prepate the dataset
    final J48 classifier = (J48)SerializationHelper.read("./j48.model");
    final var stringToWordVector =
        (StringToWordVector)SerializationHelper.read("./stwv.filter");
    tests = Filter.useFilter(tests, stringToWordVector);
    tests.setClassIndex(0);

    // Classify
    for (int i = 0; i < tests.numInstances(); i++) {
      final var score = classifier.classifyInstance(tests.instance(i));
      final var prediction = tests.classAttribute().value((int)score);
      System.out.println(prediction);
    }
  }
}
```

6.0.6. Posibles mejoras a futuro

- Por un lado seria interesante ver como usando $IDFTransform/TFTransform$ podriamos alterar los resultados.
- Debido a que los datos no estan balanceados usar el accuracy no es la mejor idea y se debe hacer con cuidado.

6.0.7. Conclusión

Gracias a los resultados vimos que es posible crear un algoritmo con la ayuda de los árboles de decisión que nos permite clasificar con un gran desempeño la urgencia de un "support ticket".

Bibliografía

[1] *Aidan Wilson*, Sep 29, 2019
https://towardsdatascience.com/a-brief-introduction-to-supervised-learning

[2] *Diego Lopez Yse*, Apr 17, 2019
https://towardsdatascience.com/the-complete-guide-to-decision-trees