Trabajo Práctico (TP6-2)

Filtrado de SPAM

Javier Herrer Torres (NIP: 776609)

Inteligencia Artificial
Grado en Ingeniería Informática

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
Curso 2020/2021

1. Objetivos

1. Entender cómo funciona un sistema de filtrado de spam en correos electrónicos.
2. Aplicar un clasificador de Bayes ingenuo a un problema real.
3. Diseñar e implementar un sistema de detección de spam en Python.
4. Evaluar el sistema implementado en bases de datos públicas.

2. Metodología

2.1. Datos

Se ha inicializado la estructura para un **modelo de bolsa de palabras normalizado** conteniendo la frecuencia de aparición en lugar del número de apariciones mediante:

```
class sklearn.feature_extraction.text.TfidfVectorizer(...)
```

2.2. K-fold Cross-Validation

El entrenamiento del clasificador se ha realizado utilizando:

```
class sklearn.model_selection.KFold(...)
```

y, posteriormente, se han analizado los resultados para elegir el clasificador que funciona mejor mediante:

```
sklearn.model_selection.cross_val_score(...)
```

2.3. Clasificadores

Se han usado dos tipos de clasificadores:

- **NB Multinomial**: el mail se representa con una bolsa de palabras entera, donde cada x_i contiene la frecuencia de aparición de una palabra en un mail (para una mayor invarianza al tamaño de los correos).
- **NB Bernoulli multivariable**: el mail se representa con una bolsa de palabras binaria, donde cada x_i modela la presencia/ausencia de una palabra.

2.4. Suavizado de Laplace

Mediante esta técnica conseguimos simular haber visto k veces el vocabulario entero en spam y ham. Aplicando la siguiente ecuación:

$$P_{LAB,k}(w_i|s) = \frac{c(w_i, s) + k}{c(s) + k|W|}$$

Donde $c(w_i, s)$ son las veces que aparece w_i en los spam, $c(s)$ el número de palabras en los spam, y $|W|$ el número de palabras en el vocabulario.

Para valores altos de k se obtienen resultados irrelevantes. Por eso se ha optado por emplear valores reales de k entre 0 y 1.

El valor de k se debe especificar en el atributo `alpha` (por defecto a 1.0) de la clase `MultinomialNB`.

2.5. Bolsa de n-gramas

Un n-grama es una secuencia ordenada de n palabras en un texto. Relaja en parte la suposición de independencia de las palabras del clasificador de Bayes ingenuo. Se debe especificar en el atributo `ngram_range` de la clase `TfidfVectorizer`.

A pesar de que los modelos de n-gramas son muy relevantes para el entendimiento de lenguaje, **no aportan mucha mejora** en el caso de clasificación de spam.

3. Resultados

3.1. Entrenamiento del clasificador

Para evaluar el mejor clasificador obtenido se han empleado las siguientes métricas:

- **Curva precision-recall**: gráfico bidimensional que resulta de unir los pares precision-recall en diferentes experimentos de clasificación cambiando el umbral de clasificación τ.
- **F1 score**: media armónica entre precisión y recall.
- **Matriz de confusión**: tabla en la que las filas se refieren a las categorías reales y las columnas a las categorías predichas, y en cada celda se almacena la cuenta o la frecuencia de cada caso.

Mediante el siguiente fragmento de código se han obtenido las puntuaciones de *accuracy* de todas las combinaciones de parámetros:

```
classifiers = [MultinomialNB(), BernoulliNB()]
clf_names = ["MultinomialNB", "BernoulliNB"]
vectorizers = [CountVectorizer(), TfidfVectorizer()]
vec_names = ["CountVectorizer", "TfidfVectorizer"]
```

```
laplace = [0.1, 0.25, 0.5, 0.75, 1]
ngrams = [(1,1), (1,2)]
ngrams_names = ["Unigrams", "Unigrams and bigrams"]
scores = ["accuracy", "f1"]

k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
for clf, clf_name in zip(classifiers, clf_names):
    for vec, vec_name in zip(vectorizers, vec_names):
        X = vec.fit_transform(mails)
        X_test = vec.transform(mails_test)
        for k in laplace:
            clf_laplace = clf.set_params(alpha = k)
            for ngram, ngram_name in zip(ngrams, ngrams_names):
                vec.set_params(ngram_range = ngram)
                start = time.time()
                print ("------------------------")
                print ("Classifier: ", clf_name)
                print ("Vectorizer: ", vec_name)
                print ("Laplace:    ", k)
                print ("Ngram:      ", ngram_name)
                for score in scores:
                    print ("%-10s %s" %(score,
                            cross_val_score(clf_laplace, X, y,
                                            scoring=score,
                                            cv=k_fold).mean()))
                print ("Time : ", time.time() - start, "\n")
```

Estos han sido los 4 resultados más altos ordenados de mejor a peor (la traza completa de ejecución con todas las combinaciones se encuentra en el fichero adjunto `trazaParametros.txt`):

```
------------------------
Classifier:  MultinomialNB
Vectorizer:  TfidfVectorizer
Laplace:     0.1
Ngram:       Unigrams
accuracy   0.9891759659803687
f1         0.9881144292282482
Time :  1.1340053081512451

------------------------
Classifier:  MultinomialNB
Vectorizer:  TfidfVectorizer
Laplace:     0.1
Ngram:       Unigrams and bigrams
accuracy   0.9891759659803687
f1         0.9881144292282482
```

```
Time :  1.129936695098877
------------------------
Classifier:  MultinomialNB
Vectorizer:  TfidfVectorizer
Laplace:     0.25
Ngram:       Unigrams
accuracy   0.9883460578877395
f1         0.987175063601275
Time :  1.138953447341919

------------------------
Classifier:  MultinomialNB
Vectorizer:  TfidfVectorizer
Laplace:     0.25
Ngram:       Unigrams and bigrams
accuracy   0.9883460578877395
f1         0.987175063601275
Time :  1.1100316047668457
```

3.2. Evaluación del mejor clasificador con los datos de test

Se ha realizado la evaluación de la mejor configuración con los **datos de test** obteniendo las siguientes métricas:

- F1 score: 0,9897936543155091
- Matriz de confusión: figura 1.
- Curva de presision-recall: figura 2.

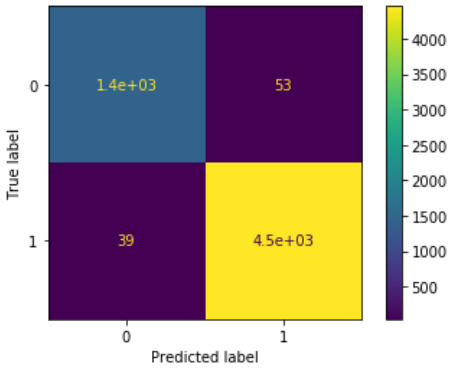

Figura 1: Matriz de confusión

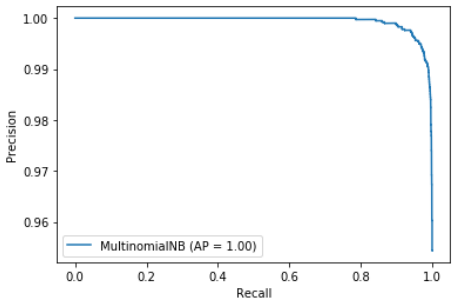

Figura 2: Curva de presision-recall

4. Conclusiones

Se comprueba por tanto que los mejores parámetros son:

- Clasificador multinomial.
- Modelo de bolsa de palabras normalizado.
- Valores reales pequeños (cercanos al 0) del suavizado de Laplace.

Por otro lado, el modelo de bolsa de palabras (con o sin bigramas) no tiene ningún efecto (es indiferente).

Tras la lectura del estudio Spam Filtering with Naive Bayes y la experimentación con los conjuntos de datos, se han extraído varias conclusiones.

Dado que la tolerancia de la mayoría de los usuarios a la hora de equivocarse en la clasificación de los correos electrónicos es muy limitada, observando las métricas se ha seleccionado un umbral de un 20 % para el clasificador de spam. Es decir, no debería de equivocarse en más de un 20 % clasificando los mensajes de spam.

Además, algunos conjuntos de datos, como Enron4, pueden ser «más fáciles» que otros, como Enron1. No parece que exista ninguna justificación para estas diferencias en términos del ratio ham-spam o la fuente de spam usada en cada conjunto.