Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2613 — Inteligencia Artificial — 2' 2019

Tarea 3 – Detección de Sarcasmo en Texto
Entrega Viernes 22/11, 23:59

1 Objetivo

En esta tarea usted tendrá la oportunidad de utilizar librerías de Python para experimentar con dos de los algoritmos de aprendizaje de máquina que vimos en clases: Máquinas de Vectores de Soporte (SVMs) y Redes Neuronales. En particular, deberá probar ambos algoritmos en una tarea de clasificación binaria, consistente en detectar si una oración fue escrita en tono sarcástico o no. A nivel general los objetivos a cumplir son:

- Preprocesar los datos para dejarlos en un formato adecuado para la aplicación de los algoritmos de clasificación.
- Experimentar con librerías que permitan utilizar los clasificadores SVM y Red Neuronal.
- Analizar hiperparámetros de ambos clasificadores para ver su impacto en el rendimiento de éstos.

2 Dataset

Para esta tarea utilizará el dataset de dominio público *"News Headlines for Sarcasm Detection"*[1], al cual puede acceder desde el siguiente link:
`https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection`

Este dataset fue construido extrayendo titulares "no sarcásticos" desde un sitio web de noticias llamado *HuffPost*, y titulares "sarcásticos" desde un sitio con noticias escritas intencionalmente en tono sarcástico llamado *TheOnion*. El set de datos cuenta con más de 25.000 registros, para los cuales existen dos campos relevantes:

- **headline:** texto correspondiente al titular de cada comentario.
- **is_sarcastic:** rótulo binario que indica si el titular es sarcástico o no.

3 Preprocesamiento del Dataset

Su primera labor consistirá en transformar el dataset de comentarios, inicialmente en un espacio de textos, a una representación o espacio de características numérica. Para esto se sugiere utilizar alguna de las siguientes representaciones:

- **Codificación BoW:** esta representación consiste en crear un histograma con la frecuencia de ocurrencia de las distintas palabras que aparecen en el texto que se desea codificar. Así, la dimensionalidad del vector de características es igual al número de palabras distintas que aparecen en el set de entrenamiento.
- **Codificación word2vec:** esta representación consiste en una red neuronal pre-entrenada que permite codificar cada palabra de texto en un espacio de características de 300 dimensiones, de ahí su nombre Word-To-Vector o simplemente word2vec.

3.1 Ejemplo de uso de BoW

```
from sklearn.feature_extraction.text import CountVectorizer

train_sentences = ['this is a sentence','this is another sentence']
test_sentences = ['this is a test sentence','this is another test sentence']
vectorizer = CountVectorizer(train_sentences)
X_train = vectorizer.fit_transform(train_sentences)
X_test = vectorizer.transform(test_sentences)
```

3.2 Ejemplo de uso de word2vec

Para ejecutar este ejemplo, primero deberá descargar el modelo preentrenado para word2vec desde el siguiente link:
`https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing`

En caso de utilizar Google Colab, puede descargarlo directamente con estos comandos:

```
!pip install gdown
!gdown --id 0B7XkCwpI5KDYNlNUTTlSS21pQmM
!gunzip GoogleNews-vectors-negative300.bin.gz
```

El siguiente ejemplo muestra cómo usted puede convertir una oración a un vector utilizando word2vec:

```
import gensim, numpy

# creamos el modelo word2vec a partir del archivo binario
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)

# separamos el texto en palabras y convertimos cada palabra en vector usando word2vec
sentence = 'this is an example sentence with unknown words like adfdesfdutd'
sentence_words = [word for word in sentence.split() if word in model.vocab]
vectors_sequence = [model.wv[word] for word in sentence_words]

# convertimos la secuencia (de largo variable) de vectores en un unico vector.
# en este caso usamos un simple promedio, pero si lo desea puede usar otras variantes mas sofisticadas.
sentence_vector = numpy.mean(vectors_sequence,axis=0)
```

3.3 Opcional: Eliminación de Stop-Words

En tareas de análisis de texto es común eliminar ciertas palabras poco informativas. Por ejemplo, en español las palabras “en”, “a” o “y” generalmente no aportan mucho a tareas de clasificación pues es común

que éstas aparezcan indistintamente en frases de cualquiera de las clases. Por esta razón, muchos modelos se ven beneficiados de que estas palabras hayan sido removidas, permitiéndoles enfocar sus esfuerzos en analizar las demás palabras. El siguiente ejemplo muestra una forma de eliminar Stop-Words de una oración:

```
from spacy.lang.en.stop_words import STOP_WORDS

sentence = 'this is an example sentence with stop words'
clean_sentence = ' '.join([word for word in sentence.split() if word.lower() not in STOP_WORDS]
```

3.4 Partición del Dataset

A partir de los datos usted deberá crear en forma aleatoria conjuntos de entrenamiento, validación y test, en proporciones de 70/15/15. Para esto puede utilizar la funcionalidad "train_test_split" disponible en la librería sklearn :
`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

4 Implementación de Clasificadores

Para implementar los clasificadores podrá ocupar la librería `sklearn` de Python. Ésta contiene implementaciones sencillas de utilizar para varios modelos de aprendizaje de máquina. En particular, ustedes deberán usar las clases:

- **SVC, implementación de un SVM:**
  `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

- **MLPClassifier, implementación de una red neuronal multicapa:**
  `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

Cada uno de estas clases tiene asociado un método `fit(x, y)` que toma una lista de inputs y una lista de outputs esperados y efectúa el entrenamiento de acuerdo con los hiperparámetros con los cuales se instanció la clase.

5 Actividades

5.1 Actividad 1

Haga un análisis de las palabras más utilizadas en el dataset. En particular, construya un histograma de frecuencia, en el cual el eje X contenga las palabras ordenadas según valores decrecientes de frecuencia. Comente acerca de la forma de este histograma.

5.2 Actividad 2

Mencione la estrategia que usó para preprocesar los datos (ya sea BoW, word2vec u otra de su elección). Explique a nivel general en qué consiste la transformación aplicada.

5.3 Actividad 3

Entrene el clasificador SVM, varíe el valor de C y cambie el tipo de kernel a ocupar entre *'rbf'*, *'poly'* y *'linear'*. ¿Cómo afectan estos cambios al rendimiento del clasificador en el conjunto de entrenamiento y validación? Reporte en una tabla el rendimiento final de todos los modelos que probó tanto en training como en validación.

5.4 Actividad 4

Elija el mejor clasificador SVM según el análisis de la actividad anterior. Use el conjunto de validación para determinar esto, comentando el criterio que utilizó para hacer la selección. Evalúe este modelo en el conjunto de test, reportando rendimiento y mostrando ejemplos (textos) donde el clasificador entregue buen resultado y ejemplos donde entregue una clasificación errónea. Comente brevemente sus principales observaciones acerca de sus resultados.

5.5 Actividad 5

Respecto a las redes neuronales, entrénelas por 30 épocas, use el optimizador *'adam'* y experimente con la cantidad de capas y el tamaño de éstas. Use el conjunto de validación para determinar una estructura adecuada. Grafique la pérdida del conjunto de entrenamiento y la exactitud en el conjunto de validación de las distintas estructuras que utilizó en su evaluación. Comente brevemente sus resultados.

5.6 Actividad 6

Elija la mejor red neuronal en base al conjunto de validación. Reporte la exactitud en el set de test. Comente brevemente sus resultados.

5.7 Actividad 7

Presente ejemplos donde el clasificador elegido en la actividad anterior entrega buenos resultados y ejemplos donde entrega malos resultados. Compárelos con los resultados del clasificador SVM. ¿Observa algún patrón o se comportan de manera similar? Comente.

6 Formato de Entrega

Como resultado de su tarea deberá entregar un informe que incorpore lo siguiente:

- Un Jupyter Notebook con todas las celdas de código ejecutadas y el resultado claramente visible.
- Si una celda no está ejecutada o presenta errores se asumirá que no se ejecutó y no se evaluará los resultados correspondientes a esa celda.
- Las respuestas a cada una de las actividades planteadas deben ser explícitas y estar escritas en la sección correspondiente a cada actividad. Dado que cada actividad será corregida por separado, no puede asumir que para la corrección de cierta actividad, el ayudante leerá la respuesta a otra actividad para encontrar la respuesta.
- El documento debe ser subido a su **GitHub**. Debe estar subido **antes de la fecha y hora límite: 22/11, 23:59** .

7 Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.

References

[1] Rishabh Misra and Prahal Arora. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*, 2019.