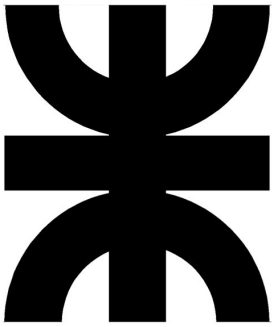

UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL CÓRDOBA

Ingeniería en Sistemas de Información

Cátedra de Inteligencia Artificial

Sobre la automatización de la obtención de información relevante a la ciberseguridad a través de Twitter

Docentes

Destefanis, Eduardo Atilio

Paez, Nancy Del Valle

Arcidiácono, Marcelo José María

Autores

Brond, Matías

Filardo, Juan Ignacio

Silva, Federico

Curso 5K2

2018

Sobre la automatización de la obtención de información relevante a la ciberseguridad a través de Twitter

Brond, Matías - Filardo, Juan Ignacio - Silva, Federico
Universidad Tecnológica Nacional, Facultad Regional Córdoba

Abstract
Se estudian distintas disciplinas de la Inteligencia Artificial como el Procesamiento del Lenguaje Natural, sentiment analysis y Naive Bayes Classifier, para analizar publicaciones realizadas en la red social Twitter y determinar su relevancia con respecto a la seguridad informática.
Estas publicaciones se conocen como «tweets» y son aquellas que formarán una parte esencial en nuestro análisis.
El presente trabajo muestra el preprocesamiento realizado al texto de los tweets antes de usarlos para entrenar un clasificador bayesiano ingenuo y generar así un modelo.
Este modelo será capaz de poder dar una clasificación binaria a cada tweet, y etiquetarlo como relevante o irrelevante en función de las palabras que contiene.
El objetivo final de poder discernir entre estas dos clases brindarle a los Analistas de Seguridad mayor fiabilidad en cuanto a la información que se les presenta, ya que por su formato, Twitter permite obtener información relacionada a ciberseguridad a través de noticias en tiempo real que son cortas, fáciles de procesar, y son difundidas incluso con anterioridad a que esa información se encuentre disponible en bases de datos especializadas.

Palabras Clave
Procesamiento del Lenguaje Natural - Tweets - Machine Learning - Naive Bayes - Ciberseguridad - Python - Sentiment Analysis

Introducción

El Procesamiento del Lenguaje Natural o *Natural Language Processing* (de aquí en más PLN) se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio de lenguajes naturales. Las lenguas humanas pueden expresarse por escrito (texto), oralmente (voz) y también mediante signos. Naturalmente, el PLN está más avanzado en el tratamiento de textos, donde hay muchos más datos y son más fáciles de conseguir en formato electrónico.

Una rama del PLN es el *sentiment analysis*, un mecanismo que permite identificar y extraer información subjetiva de los recursos. Es una tarea de clasificación masiva de documentos de manera automática, en función de la connotación positiva o negativa del lenguaje utilizado en el documento. Estos procesos suelen basarse en relaciones estadísticas y de asociación de palabras, no en análisis lingüístico propiamente dicho.

Una herramienta que utiliza estas relaciones estadísticas es clasificador Bayesiano ingenuo es un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales. Es a causa de estas simplificaciones, que se suelen resumir en la hipótesis de independencia entre las variables predictoras, que recibe el apelativo de ingenuo.

Un clasificador de Bayes ingenuo[1] asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica, dada la clase variable.

Para este caso de estudio el clasificador asume que un tweet sea relevante o irrelevante

Desarrollo

Se toma la idea propuesta por el competición WCCI[2] 2018, concurso organizado por el proyecto DiSIEM[3] de la Unión Europea H2020. El objetivo de este proyecto es mejorar las capacidades de los sistemas de Información de Seguridad de Gestión de Eventos utilizando tecnología relacionadas con la diversidad.

Los organizadores de la competencia proveen un dataset con 6694 tweets (tanto su texto, sus metadatos, y una etiqueta, explicada más abajo) con el objetivo de que los competidores creen un algoritmo de clasificación con dos principales propósitos:

Maximizar la cantidad de información relevante presentada a los analistas de seguridad.

Minimizar la cantidad de información irrelevante presentada a los analistas.

La idea es analizar los tweets, y poder clasificarlos en dos clases (relevantes o no), en función de las amenazas de seguridad que contiene.

Figura 1: Nube de palabras que aparecen en tweets relevantes.

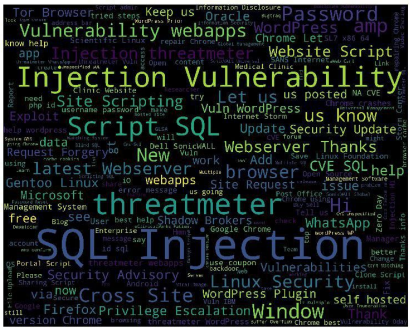

Figura 2: Nube de palabras que aparecen en tweets irrelevantes.

Nota: la solución descrita aquí debajo está explicada paso por paso y de forma detallada en la Jupyter Notebook[4] anexa.

En pocas palabras, la solución planteada para resolver el problema se basa en un clasificador binario Naive Bayes, aunque podría utilizarse cualquier otro tipo de clasificador para estudiar las diferentes performances de cada uno.

Para comenzar a observar cómo está compuesto el dataset obtenido, se lo convierte a formato json, luego se lo carga a una estructura nativa de Python[5] (i.e. un diccionario), y utilizando la función pandas.DataFrame.from_dict() que provee la librería pandas, podemos convertir los datos a una estructura de datos conocida como DataFrame[6], que permite manipular y trabajar con datos de una forma simple, sencilla y eficaz.

A partir del DataFrame generado, se crea otro que posee únicamente los datos «text» (el texto plano del tweet), «hashtags» (los hashtags del tweet, i.e. una palabra o conjunto de palabras precedidas por el símbolo #) y «classification» (la etiqueta de esa muestra, que es igual a 1 si el tweet es relevante, y -1 si no lo es).

A continuación se utiliza una función de la librería scikit-learn[7] que nos permite mezclar y separar el corpus en un conjunto

de entrenamiento y un conjunto de prueba, que de aquí en más serán llamados training set y test set respectivamente. El 90% de los tweets se usarán para entrenar el clasificador, mientras que las pruebas se harán en el 10% restante.

Una vez separado el corpus, se deja de lado el test set y se trabaja únicamente con el training set, procediendo al procesamiento de cada uno de los tweets: se eliminan las palabras menores a 3 caracteres y stopwords como «http», «RT», hashtags y usernames. Vale la pena destacar que se realizaron dos pruebas, en la primera se eliminaron las stopwords especificadas en el conjunto de stopwords provistas por nltk. La segunda prueba se realizó teniendo en cuenta las mismas y se compararon y analizaron los resultados obtenidos.

Se procede a obtener todas las palabras de los tweets, y esto se realiza con la función *get_words_in_tweets(tweets)*, que devuelve una lista con todas ellas, repetidas en el caso de que lo estén. Una vez que tenemos esta lista, dentro de *get_word_features(wordlist)* se usa la función *nltk.FreqDist(wordlist)* para obtener la distribución de frecuencia de las palabras en el texto, y a partir de ella se retorna la lista de palabras pero sin repetir, como si fuera un set de python.

El clasificador bayesiano provisto por nltk requiere una función que «extraiga las features» de cada uno de los documentos (en este caso tweets) a analizar. En este caso, esa función es *extract_features(document)* y lo que hace es retornar un diccionario que verifica si un tweet contiene o no las palabras del corpus, E.g.:

```
{
   'contains(virus)': True,
   'contains(wordpress)': False,
   'contains(adobe)': True,
   'contains(hat)': False, ...
}
```

Para el caso de estudio actual se manipularon las palabras que componen dichos cuerpos, estas se analizan mediante un clasificador bayesiano, el cual trabaja según la probabilidad de la existencia de la palabra analizada en un texto relevante y la probabilidad de existencia de esa misma palabra en un texto irrelevante. En base a dicho análisis, se le asigna una reputación a esa palabra (1 en caso de ser relevante o -1 en caso contrario).

Resultados obtenidos

Los resultados obtenidos se dividen de la siguiente manera, según si durante el preprocesamiento se descartaron las stopwords o no, y si se incluyeron o no los hashtags como una feature separada.

Las métricas utilizadas aquí son la True Positive Rate y True Negative Rate, también llamadas sensibilidad y especificidad[8].

En el problema aquí presente, la sensibilidad mide cuántos tweets relevantes fueron clasificados como relevantes, y la especificidad mide cuántos tweets irrelevantes fueron correctamente clasificados.

La sensibilidad y especificidad variaron de la siguiente manera:

Sensibilidad	Sin stopwords	Con stopwords
Sin hashtags	0.6110	0.9344
Con hashtags	0.6849	0.8073

Tabla 1: Probabilidades de clasificación de un tweet como relevante, tanto con stopwords como sin, con hashtags y sin hashtags.

Especificidad	Sin stopwords	Con stopwords
Sin hashtags	0.8918	0.6480
Con hashtags	0.8885	0.7147

Tabla 2: Probabilidades de clasificación de un tweet como irrelevante, tanto con stopwords como sin, con hashtags y sin hashtags.

Pueden observarse aquí distintos fenómenos. Para empezar, al quitar las stopwords de los tweets, el clasificador tiende a ser más «pesimista»: clasifica más tweets como irrelevantes, pero esto provoca una marcada pérdida de la sensibilidad, con caídas de entre 18% y 22%.
Quitar los hashtags produce distintos resultados: si las stopwords se quitan la especificidad es más alta que si las stopwords se mantienen, pero la sensibilidad es más baja.
Si los hashtags son dejados en el corpus, la presencia de stopwords produce una menor especificidad pero una mayor sensibilidad.

Conclusión y Trabajos futuros

Resulta difícil encontrar un método estadístico que permita diferenciar los tweets meramente por la frecuencia de las palabras que contiene, ya que muchas palabras aparecen tanto en tweets relevantes como irrelevantes, lo cual puede observarse en las figuras 1 y 2.
Los clasificadores como el utilizado en este trabajo, al igual que todos los que utilizan Machine Learning, se basan en encontrar cualquier correlación posible en el training set, lo cual también incluye correlaciones espurias; haciendo que los clasificadores estén lejos de encontrarse libres de sesgos.
Un análisis más exhaustivo que podría realizarse para obtener mejores resultados sería usar métodos propios de sentiment analysis para encontrar significado a las palabras más allá de su frecuencia. Para ejemplificar por qué esto es un problema, basta con pensar en la frase «not relevant»: el sistema aquí planteado la procesaría como dos palabras distintas, «not» y «relevant», y no podría distinguir que el significado de la oración es negativo.

Referencias

[1] Clasificador Bayesiano Ingenuo: https://es.wikipedia.org/wiki/Clasificador_bayesiano_ingenuo
[2] Competencia WCCI 2018: http://disiem-project.eu/index.php/wcci-2018-competition
[3] DiSIEM: http://disiem-project.eu/
[4] Jupyter Notebook: https://github.com/JuaniFilardo/tp-ia/blob/master/tweets_classifier.ipynb
[5] Documentación de Python: https://docs.python.org/3/
[6] Documentación DataFrame: http://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.html
[7] Scikit-learn: http://scikit-learn.org/stable/
[8] Sensibilidad y especificidad: https://es.wikipedia.org/wiki/Sensibilidad_y_especificidad_(estad%C3%ADstica)

Datos de Contacto

Brond, Matías - matibrond@gmail.com
2954 552865 - t.me/MatiBrond

Filardo, Juan Ignacio - jifilardo@gmail.com
3543 300050 - t.me/JuaniFilardo

Silva, Federico fedenanosilva@gmail.com
351 2045980 - t.me/FedeSilva