Memoria Challenge Final

Joaquín Sánchez-Cascado

Introducción

En esta memoria se detallan las actividades realizadas por Joaquín Sánchez-Cascado, utilizando como inicio las actividades comunes explicadas en el documento de “Memoria Challenge Final”.

Objetivos

El objetivo de las tareas fue aplicar técnicas de procesado de lenguaje natural y de aprendizaje automático y desarrollar un modelo capaz de clasificar textos como ofensivos o no ofensivos a partir de un dataset etiquetado.

Tareas Realizadas

1. **Preprocesar** el texto para mejorar la calidad de los datos.
2. **Vectorizar** el dataset para poder clasificar.
3. **Entrenar** modelos de clasificación.
4. **Evaluar** el rendimiento de los modelos.
5. **Optimizar** mediante GridSearchCV para mejorar la precisión de los modelos.

Preprocesamiento del Texto

Este paso previo fue realizado por todo el grupo en conjunto en un notebook separado y a partir de él, cada integrante del grupo añadió las funciones que más convenían en función del modelo implementado.

El preprocesamiento es crucial para mejorar la calidad del texto y facilitar el entrenamiento del modelo. Las tareas realizadas fueron:

- Se eliminaron URLs y emojis del texto.
- Se corrigió la ortografía utilizando un corrector automático.
- Se eliminaron signos de puntuación.
- Se eliminaron las palabras vacías (stopwords) del inglés.

En mi caso, no utilicé eliminación de URLs ni de emojis ya que no había la suficiente cantidad (no mejoraría tanto la clasificación) para el tiempo que tardaba en realizar

esa función (se alargaba de más). Una mejora podría consistir en utilizar dichos emojis y URLs para clasificar dependiendo de estos.

Representación del Texto con Word2Vec

Se utilizó word2vec para transformar las palabras en vectores de 100 dimensiones. Word2Vec es una técnica que permite capturar relaciones semánticas entre palabras. Se entrenó un modelo Word2Vec con el conjunto de datos de entrenamiento para obtener los vectores de palabras.

Entrenamiento de Modelos

Se entrenaron tres modelos diferentes utilizando las representaciones vectoriales obtenidas con Word2Vec:

- **Regresión Logística**.
- **SVM**
- **Random Forest**.

Optimización con GridSearchCV

Se utilizó GridSearchCV para buscar los mejores hiperparámetros para cada modelo. Los parámetros ajustados incluyen la penalización (L1/L2) y la regularización y los parámetros del kernel en SVM.

Análisis y Conclusiones

Elección de Word2Vec

En todos los notebooks he decidido utilizar Word2Vec para entender cómo funciona y conocer su rendimiento para luego compararlo con los demás métodos de mis compañeros. Se ha elegido porque:

Word2Vec es un algoritmo popular utilizado para el procesamiento del lenguaje natural y la clasificación de textos. Es un enfoque basado en redes neuronales que aprende representaciones distribuidas (también llamadas incrustaciones) de palabras a partir de un gran corpus de texto. Estas incrustaciones capturan las relaciones semánticas y sintácticas entre términos.

El objetivo era comparar la efectividad con TF-IDF: TF-IDF (frecuencia de término-inversa de frecuencia de documento) y Word2Vec son algoritmos populares en el procesamiento del lenguaje natural, pero son diferentes. TF-IDF es un método simple utilizado para la representación de texto que asigna pesos a las palabras según su frecuencia y frecuencia inversa en el corpus. Por otro lado, Word2Vec es un algoritmo más complejo que aprende representaciones vectoriales de palabras basadas en su contexto en un gran corpus de texto, mostrando cómo las palabras están relacionadas semántica y gramaticalmente.

TF-IDF es más adecuado para conjuntos de datos más pequeños y Word2Vec más eficaz para conjuntos de datos más grandes y complejos. Quedando en evidencia este hecho al ver que el procesado con TF-IDF nos ha arrojado resultados mejores sobre este dataset, el cual es relativamente pequeño.

Word2Vec tiene dos modos de trabajo. El modelo CBOW (Continuous Bag of Words) predice una palabra objetivo a partir de sus palabras de contexto circundantes (dentro de una ventana (window)), utilizando todas las palabras de contexto para predecir la palabra en el medio. Por ejemplo, en la frase "El gato se sentó en la alfombra", si usamos "gato" como palabra objetivo, el modelo CBOW tomará "El", "se sentó", "en", "la", "alfombra" como contexto para predecir "gato". Este modelo es beneficioso con conjuntos de datos pequeños y es más rápido que el modelo Skip-Gram. Por otro lado, el modelo Skip-Gram predice las palabras de contexto circundantes a partir de una palabra objetivo, utilizando una sola palabra para predecir su contexto. Este modelo es eficaz con conjuntos de datos grandes y con palabras raras, y aunque es computacionalmente más complejo, ofrece ventajas en la captura de relaciones semánticas y manejo de palabras raras (el dataset contiene jergas y abreviaciones). Finalmente, se ha escogido el modelo Skip-Gram con una ventana de 5 palabras porque ha dado un mayor accuracy.

Elección de algoritmos de clasificación

He decidido dividir el trabajo realizado en tres notebooks en el que cada uno evalúa un modelo de clasificación, teniendo todos en común, como se ha comentado, el proceso de vectorización. Con esto se consigue conocer cuales son los mejores algoritmos de clasificación para word2vec.

- Logistic Regression por su buen rendimiento y fácil interpretación. La Regresión Logística es un algoritmo de clasificación lineal que se utiliza por su simplicidad y eficacia. Quería utilizar un algoritmo más simple para poder también comparar frente a otros más complejos. Finalmente, como se ha visto, no ha habido un cambio tan diferencial frente a los otros dos (0,1 menos en accuracy).

- SVM es un clasificador robusto que maneja eficazmente espacios de alta dimensión, aprovechando los embeddings pre entrenados de Word2Vec para captar patrones.

- Random Forest es un algoritmo que consiste en un bagging de árboles de decisión. Al contrario que el primero, este se ha decido usar por el insight que puede ofrecer usar un algoritmo de mayor complejidad.

Por último, la optimización con GridSearch no ha añadido mucha mejora. Previamente, se obtenía en los tres modelos de clasificación un accuracy de dos puntos menos. Tras varias pruebas, los parámetros utilizados han sido los que mejores resultados daban frente de los que se han probado.

Resultados

El modelo de SVM con los parámetros optimizados obtuvo la mejor precisión en el conjunto de prueba (0,67 de accuracy).

Debido al resultado tan parecido entre los tres modelos (de diferentes complejidades), podemos suponer que el cuello de botella se encuentra en una de las dos siguientes partes: el preprocesado o el uso del modelo word2vec para la vectorización del dataset.

Comparando estos resultados con los de mis compañeros, los cuales, como se ha mencionado, han realizado el mismo pre-procesado, puedo comprobar que efectivamente, el cuello de botella se encuentra en el uso del modelo de vectorización word2vec.

Conclusión y Futuras Mejoras

La combinación de Word2Vec con algoritmos de clasificación supervisada ha demostrado ser efectiva para la detección de discurso de odio pero no la óptima. La optimización de hiper parámetros mediante GridSearchCV ha mejorado la efectividad y la precisión del modelo pero no ha sido diferenciativo. El modelo SVM resultó ser el más preciso.

Para futuras mejoras, el uso de otro vectorizador como paso previo a la clasificación sería el aspecto a mejorar. Explorar modelos más complejos y efectivos para clasificar los resultados como Deep Learning, redes neuronales, etc… también podría mejorar. También se podría experimentar con técnicas de preprocesamiento de texto más avanzadas.

Bibliografía

[1] Classification using Word2vec. [En línea]. Disponible en: https://medium.com/@dilip.voleti/classification-using-word2vec-b1d79d375381

[2] word2vec - TensorFlow. [En línea] . Disponible en: https://www.tensorflow.org/text/tutorials/word2vec

[3] *Word2Vec and Logistic Regression*. [En línea] . Disponible en: https://www.kaggle.com/code/danielbilitewski/word2vec-and-logistic-regression

[4] *Tutorial: Word Embeddings with SVM*. [En línea]. Disponible en: https://www.kaggle.com/code/mehmetlaudatekman/tutorial-word-embeddings-with-svm

[5] *word2vec and random forest classification*. [En línea]. Disponible en: https://www.kaggle.com/code/arunava21/word2vec-and-random-forest-classification

[6] *Tune Hyperparameters for Classification Machine Learning Algorithms*. [En línea]. Disponible en: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/

Valoraciones

	Valoración	**Repetir**
Pablo Fernández	8.5	Sí
Iván Caamaño	8	Sí
David Plaza	8,5	Sí
Joaquín Sánchez-Cascado	8	