Natural Processing Text Classification

GUILLERMO GARCIA OTIN

November 2020

1 Data analysis and Data cleaning

Primero, se ha ploteado la cantidad de opiniones para cada Rating. Se puede comprobar que hay 2000 clases de cada una, es decir, el dataset esta balanceado y esto facilita la tarea. Adicionalmente, y con el objetivo de buscar diferencias entre las distintas clases, se representan las palabras más comunes en cada clase. Para hacer esto de manera que las palabras sean significativas tendremos que hacer data preprocessing: Se elimina posible ruido debido a caracteres no convencionales que no nos aportan información significativa. Seguidamente, se aplica un *Lemmatizer* que reduce las palabras teniendo en cuenta su morfología. Se ha preferido aplicar un *Lemmatizer* en lugar de *Steammer* ya que, para contar el número de repeticiones de las palabras, usando *Lemmatizer* conseguimos una menor variabilidad y podemos englobar las palabras por su significado morfológico. A continuación, se aplica un *Tokenizer* para tener las palabras en unidades fundamentales, *tokens*, que seguidamente contaremos. Por último, se elimina las *stopwords* y los signos de puntuación.

Adicionalmente se ha investigado la influencia de la separación de contracciones (e.g. *aren't* a *are not*), ya que la palabra *not* podía desempeñar un papel importante en las *ratings* negativas. Sin embargo, se ha encontrado que esta palabra aparece de forma uniforme en todas las categorías y por lo tanto no es una feature que nos permita discernir entre categorías.

Se ha encontrado que las palabras más frecuentes en todas las categorías son las mismas excepto en el caso de la categoría 5, donde las palabras *great* y *love* aparecen frecuentemente. Por lo tanto, se ha decidido no utilizar modelos que modelen las *reviews* como conjuntos de palabras sin relación (BOG, TF-IDF). En estos modelos, la similitud entre las *reviews* se calcularía mediante la distancia (por ejemplo, distancia euclídea) entre los vectores de cada *review*. A partir de ahí se podría clusterizar en 5 clases diferentes por ejemplo con K-Means. Otra alternativa sería usar LSI que descompone la matriz TF-IDF mediante SVD, ó PLSI que utiliza el *Expectation Maximization algorithm*.

2 Learning process

Como se ha argumentado anteriormente, no se van a utilizar modelos de pura estadística de palabras. En esta tarea, se propone utilizar el modelo BERT

[1], el cual se trata de un *Bidirectional Transformer* que puede aprovechar la información de izquierda a derecha y viceversa simultaneamente. Incluso ELMo, que utiliza un LSTM bidireccional, simplemente concatena la información de izquierda a derecha y de derecha a izquierda, lo que significa que la representación no puede aprovechar los contextos de izquierda y derecha simultáneamente. Además, las capas de *attention* nos aportan la flexibilidad de poder prestar atención determinadas partes de la frase. Por si no fuera poco, este modelo cuenta con un modelo *pretrained* en grandes volúmenes de texto de una manera *unsupervised*, prediciendo la siguiente frase o la siguiente palabra. En esta tarea se propone usar el modelo *pretrained* y ejecutar *fine tunning* para nuestra tarea de clasificación. El modelo ha sido implementado en la librería *Pytorch*.

3 Models validation

3.1 Validation method

Se ha decidido usar un *Holdout Method*, es decir, el dataset ha sido dividido en tres partes: train, validation y test (0.7, 0.15, 0.15). Se ha decidido no hacer *Cross validation* ya que se ha considerado que los comentarios de una categoría no tienen excesiva varianza y, por lo tanto una única partición aporta resultados significativos.

3.2 Evaluation metric

Para la evaluación del modelo se han usado tres métricas diferentes. Las siguientes abreviaciones son usadas: TP= True positive, FP = False positive, TN = True negative, FN = False negative.

- Precision: La fracción de los elementos seleccionados que es relevante. Precision = TP / (TP + FP)

- Recall: La fracción de los elementos relevantes que ha sido seleccionada . Recall = TP/(TP + FN)

- F1-Score: La media armónica de precision y recall. F1-Score = 2/(1/Precision) + (1/Recall))

Adicionalmente se calcula la *confusion matrix* para ver las categorías que confunde entre sí.

3.3 Training and testing accuracies

Se obtienen unos resultados en el train set de: Precision: 0.9604, Recall: 0.9599 F1-Score: 0.9599.

En el test set se ha obtenido : Precision: 0.9578, Recall: 0.9567 F1-Score: 0.9567.

4 Final summary

Con más datos el modelo tendría mejores resultados. Por otro lado, con más tiempo se puede prestar más atención al *hyperparameters tunning* y obtener unos resultados mejores.

References

[1] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.