Módulo de taller

Humor computacional: Anotación de corpus y clasificación de tweets en español

Autor:
Arturo Collazo - arturo.collazo@fing.edu.uy

Supervisores:
Luis Chiruzzo -luischir@fing.edu.uy
Santiago Góngora -sgongora@fing.edu.uy

Introducción

En el marco de la edición 2021 de la competencia iberoamericana sobre humor computacional HAHA, se presenta un taller con el propósito de reconocer mecanismos y objetos (targets) de humor en tweets recopilados de cuentas humorísticas en español.

Los mecanismos pertenecen a una y solo una de doce posibles categorías, mientras que los objetos pueden pertenecer a una, varios o ninguna categoría.

Las técnicas a emplearse son de libre elección y contemplan desde un enfoque basado en reglas a uno con redes neuronales recursivas.

El enfoque de este informe busca construir el clasificador a partir de redes neuronales y el empleo de *embeddings.*

Preprocesamiento

La etapa de preprocesamiento consta de la depuración de cada una de las instancias*,* su codificación y posterior salida en un tamaño configurado, definido como la mediana de la cantidad de palabras en un tweet.

Curación

La curación de cada tweet consiste en remover caracteres indeseados (arroba, comillas, signos de puntuación, saltos de línea, caracteres unicode, entre otros), y espacios duplicados. Luego, se obtiene la mediana de la cantidad de palabras de cada tweet.

Codificación y estandarización

Utilizando embeddings, se codifican las palabras de los tweets curados, obteniendo una salida de enteros. Una vez codificados, a aquellos tweets con una cantidad de palabras mayor a la mediana, se los trunca, mientras que a aquellos con una cantidad menor, se les agrega relleno.

Procesamiento

El enfoque seleccionado para la clasificación de mecanismos es vía redes neuronales recursivas, donde los parámetros y la arquitectura se determinan a partir de una búsqueda no exhaustiva.

Línea base

Para determinar la línea base de la tarea, se emplea un clasificador aleatorio, que retorna una medida F1 (macro) de 0,07440052832082. Donde la estrategia para generar predicciones respeta la distribución de clases del juego de datos.

Búsqueda de parámetros

El alcance de la búsqueda de parámetros se limita a las funciones de activación, recursión, cantidad de unidades en las capas ocultas y la cantidad de épocas máxima y de paciencia. Mientras que las arquitecturas exploradas son: dos capas de tipo GRU, dos capas de LSTM, una capa GRU seguida de una LSTM y finalmente, una capa LSTM seguida de una GRU.

Todas las configuraciones tienen una capa de *embedding* inicial, una capa densa con doce unidades (cada una referenciando a una categoría), función de pérdida categorical y optimizador adam.

La búsqueda se realiza de forma iterativa y la salida de interés es la métrica F1 (macro), dado que el conjunto de datos está desbalanceado.

Pseudocódigo

```
for activation in activation functions
  for recurrent in recurrent functions
    for units in layer units
      generate model <- GRU GRU
      generate model <- GRU LSTM
      generate model <- LSTM GRU
      generate model <- LSTM LSTM
    end
  end
end
```

Elección de la arquitectura

A partir de reiteradas ejecuciones sobre el pseudocódigo se recopilaron distintos valores de la métrica F1 (macro), donde se evidencia que todos los experimentos que tuvieron a la función sigmoid en el papel de activación, fueron los de mayor performance dentro de su ‘grupo’.

Cada experimento finaliza tanto por alcanzar la cantidad máxima de *epochs,* o por alcanzar la cantidad máxima de *patience* sin mejoras en las métricas. En ambos casos, se almacena (o retrocede) la configuración con mejor *performance.*

Experimentos

GRU GRU

- 25 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,119252748593	0,141336729143	0,125035218235	**0,146874583431**

- 35 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,141701500184	0,136535000081	0,144865729418	**0,155260051102**

- 45 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,140860715359	0,142361384264	**0,155708731781**	0,152734598287

GRU LSTM

- 25 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,131156912998	0,136535157148	0,112363270285	**0,139811308061**

- 35 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	**0,138987952338**	0,129589750195	0,125175787508	0,131260598077

- 45 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	**0,152683599823**	0,131673290226	0,132900659296	0,148472113108

LSTM GRU

- 25 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	**0,140532747718**	0,134255715640	0,124408811821	0,121134310523

- 35 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	**0,147509120553**	0,135740523159	0,141501431891	0,128794356724

- 45 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,155453711761	0,127074629938	**0,160999177679**	0,129901749580

LSTM LSTM

- 25 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,120126693146	0,120936605655	**0,138128656409**	0,117579765032

- 35 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,133228081946	0,133586551705	**0,138002216112**	0,126804614805

- 45 neuronas

	Rec: elu	Rec: relu	Rec: sigmoid	Rec: tanh
Act: sigmoid	0,137126295589	**0,160888293670**	0,149288023613	0,13442312302

Elección del modelo final

El modelo final seleccionado para clasificar ejemplos tiene una capa de embeddings como se mencionó anteriormente, dos capas ocultas, LSTM y GRU con 45 neuronas cada una, y una densa de 12 neuronas, donde todas las capas utilizan como función de activación y recursión (si corresponde) sigmoid.

Conclusión

La construcción del clasificador a partir de la estrategia utilizada no es óptima. Si bien se obtienen mejores resultados en la línea base, tiene una performance que se considera baja. Las causas pueden responder a dos tópicos: naturaleza de los datos o metodología de construcción del clasificador.

El juego de datos con el que se entrena está desbalanceado, y la cantidad de categorías es amplio, propiedades que afectan la tarea. Además, sin incursionar demasiado en la lingüística, los chistes tienden a ser ambiguos (donde reside la gracia) y se emplean constantemente modismos propios de una región, ciudad o país lo que podría afectar el uso del *embeddings*.
Por otra parte, la búsqueda de parámetros para el clasificador no es exhaustiva, y pueden existir otras configuraciones con mejores resultados. Incluyendo: *dropou*t, *kernel_initializer,* entre otros, o directamente otras arquitecturas como son las redes convolucionales que han sido adoptadas para la clasificación de texto.

Acerca de futuras iteraciones sobre esta tarea, puede considerarse la construcción de clasificadores específicos de categorías (con respuesta binaria) que trabajen en conjunto podrían ser de utilidad, ya que aprenderían con mayor precisión que un clasificador genérico la ‘estructura’ de una categoría y la ya mencionada adopción de otros enfoques, inclusive aquellos basados en reglas.

Bibliografía

- https://keras.io/api/
- https://scikit-learn.org/stable/
- https://www.researchgate.net/profile/Luis-Chiruzzo/publication/335429672_Overview_of_HAHA_at_IberLEF_2019_Humor_analysis_based_on_human_annotation/links/5fa6c6c3458515157bf4ac5d/Overview-of-HAHA-at-IberLEF-2019-Humor-analysis-based-on-human-annotation.pdf
- https://medium.com/analytics-vidhya/understanding-nlp-keras-tokenizer-class-arguments-with-example-551c100f0cbd
- https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/