Etiquetado de partes tóxicas

Alvarado A. Morán Óscar oaam3121@gmail.com
Dante Bermúdez Marbán bermudezmarbandante@gmail.com
Martín García Avendaño firemikael12@gmail.com

Resumen

En el presente trabajo se plantea la problemática de detección de tópicos tóxicos, presentado en el SemEval 2021. Después de realizar un preprocesamiento a lo que se marcó como tóxico se probaron tres modelos para el etiquetado: modelos ocultos de Markov, perceptrón promediado y una red neuronal recurrente LSTM. Se llegó a que el mejor modelo para esta tarea fue el perceptrón.

1 Introducción

1.1 Cadenas de Markov

Una cadena de Markov indica la probabilidad del siguiente valor que puede tomar una variable aleatoria con respecto a un conjunto de valores dados. Una cadena de Markov hace una muy fuerte suposición de que el siguiente valor en la secuencia depende solamente del valor actual de esta, los valores pasados no tienen un impacto en el futuro excepto del estado actual. Es como si tratásemos de predecir el clima de mañana basados en el clima de hoy pero sin ver el clima de ayer.

Las cadenas de Markov por su naturaleza resultan particularmente útiles en el Procesamiento de lenguaje natural (PLN), especialmente en el etiquetado de palabras ya que si por ejemplo se tiene la frase: “En esta vida hay que plantar un árbol” la palabra ”plantar” es un verbo y ahora queremos saber si la siguiente palabra en la oración es un verbo, un sustantivo o alguna otra parte del discurso, con solo un poco conocimiento del lenguaje cualquier persona podría deducir que es mucho más probable que la siguiente palabra sea un sustantivo en lugar de un verbo, si se traslada esta idea aplicando cadenas de Markov podemos asumir que la etiqueta de una palabra X_{n+1} depende de la etiqueta de la palabra X_n

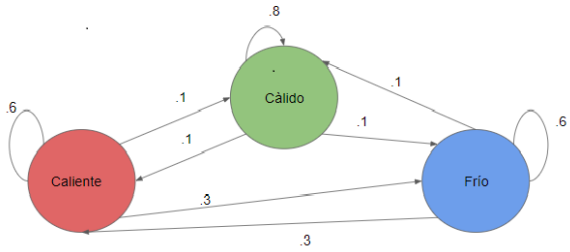

Figura 1: Cadena de Markov.

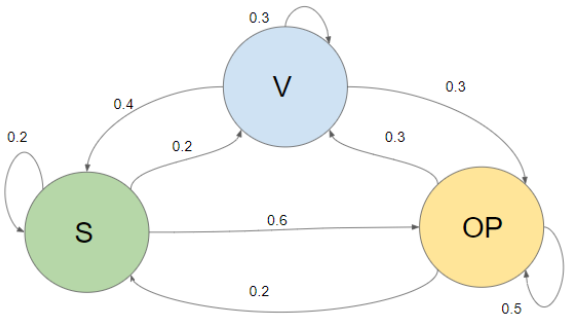

Figura 2: Palabras como grafo.

El gráfico anterior representa a la oración como una secuencia palabras y a éstas como una secuencia de eventos que pueden ocurrir en donde cada borde representa un estado que puede adquirir el modelo, es decir, la siguiente palabra de la oración puede ser un verbo (V), un sustantivo(S) o algún otra parte del discurso (OP). Los bordes del gráfico tienen una probabilidad de transición asociada, es decir, la probabilidad de que la siguiente palabra tome ese estado. Retomando el principio de Markov donde la probabilidad de un estado solo depende del estado anterior y usando la siguiente frase “Corran del perro”, si estamos en la palabra “Corran”, la probabilidad de que la siguiente palabra “perro” sea un sustantivo solo depende de la palabra “Corran”, que es un verbo por lo que la probabilidad de que “perro” sea un sustantivo es de 0.4, de que sea otro verbo de 0.3 y de que sea

otra parte del discurso es de 0.3, estas diferentes posibilidades de que se adquiera un estado pueden colocarse en una matriz de transición.

1.1.1 Matriz de transición

Una Matriz de Transición es una matriz de n por n, siendo n el número de estados que puede adquirir una palabra. Cada fila de la matriz representa la probabilidad de transición de un estado a otro tomando en cuenta que todas las probabilidades deben sumar a uno. Cuando no se tiene una palabra anterior como referencia, como es el caso cuando se inicia una frase, se puede incluir un estado inicial donde la probabilidad es asignada por uno mismo. Teniendo ahora una matriz de $(n+1) \times n$.

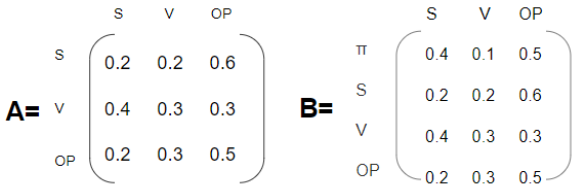

Figura 3: Matriz **A**, matriz sin un estado inicial basado en la figura 2, Matriz **B** matriz con un estado inicial basado en la figura 2

1.2 Modelo oculto de Markov

El modelo de Markov oculto se compone de un conjunto de estados que este puede adquirir (Toxico-no toxico), un matriz de transición (La probabilidad de adquirir un estado con base a otro), una probabilidad de estado inicial (La cual se define como la probabilidad de que se comience en un estado X), una secuencia de probabilidades de emisión cada una expresa la probabilidad de que se genere un estado a partir de una observación, son con estas observaciones con las que se llena la matriz de transición (Martin, 2020).

1.3 Perceptrón

Formalmente como *Averaged Perceptron*, es un modelo desarrollado por Matthew Honnibal que utiliza el concepto del perceptron para etiquetar secuencias (originalmente para etiquetado gramatical) (Honnibal, 2013).

Primeramente el modelo hace un preprocesamiento de los datos

- Todo a minúsculas
- Números entre 1800 a 2100 se representan como !YEAR
- Otros números se representan como !DIGITS

Luego, se extraen las características, que consisten en obtener la palabra actual, su contexto, sufijos, etc. Se almacenan como conjuntos ya que es más efectivo que un vector que podría resultar disperso.

Los pesos se representan como diccionarios de diccionarios que asocia tuplas de características y clases a su respectivo peso.

Para aprender los pesos correspondientes, se realiza el siguiente procedimiento iterativamente:

1. Obtener un nuevo par (características, tag)
2. Obtener predicción del tag con los pesos actuales
3. Si la predicción está mal, se le suma uno a los pesos asociados al tag correcto y restamos uno a los pesos asociados al tag predicho

Los pesos finales no son los de la última iteración, sino el promedio de todos los pesos. Esto se hace debido a la naturaleza voraz (greedy) del algoritmo: le pondrá mucha atención a las pocas oraciones que esté etiquetando mal, entonces, promediando es una manera de conservar todo el trabajo que se hizo en el entrenamiento sin arruinarlo por unos ejemplos.

1.4 LSTM

De las siglas en inglés *Long Short Term Memory* son un tipo de redes neuronales recurrentes que fueron desarrolladas para lidiar con el problema del grandiente desvaneciente, es decir, que el gradiente va teniendo valores tan pequeños que puede incluso detener a la red de seguir entrenando.

La clave de estas redes es el estado de una celda que funciona como una especie de "memoria".

A cada elemento de la secuencia en el tiempo t, se le aplican las siguientes operaciones (PyTorch, 2019)

$$i_t = \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}\right) \quad (1)$$
$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right) \quad (2)$$
$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) \quad (3)$$
$$o_t = \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}\right) \quad (4)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

Donde $\odot$ es la multiplicación elemento a elemento, también conocido como producto de Hadamard, σ es la función sigmoide

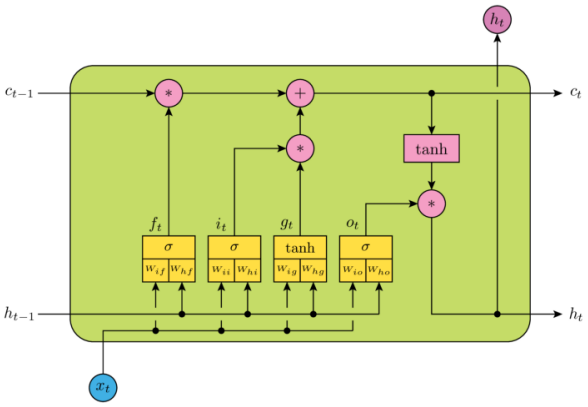

Figura 4: Visualización de una celda LSTM

En la figura 4 se ve una visualización de las operaciones mencionadas.

Intuitivamente

- f_t es la compuerta de olvido, y tiene como función decidir que tanto de la información proporcionada anteriormente y de la nueva entrada debe conservar. Como hay una sigmoide, valores cercanos a 0 significan que la mayoría de la información no es relevante y que se olvidará para actualizar la memoria de la red

- i_t es la compuerta de entrada, mientras que g_t es el candidato del nuevo estado de la memoria. i_t decide que información es relevante de g_t.

- c_t es la actualización del estado de la memoria.

- o_t es la compuerta de salida, y es lo que determina el valor del estado oculto en el siguiente bloque LSTM junto con el estado actual de la memoria.

2 Metodología

2.1 Datos

Los datos analizados tienen un formato tal como se muestra en la Tabla 1 donde la lista de números en la columna *SPANS* se refiere a índices del texto de la columna *TEXT*, que en la tabla se muestran en color rojo las palabras a las que se refiere como tramos tóxicos. En el último renglón mostrado en dicha tabla se muestra una lista vacía dado que los evaluadores del texto optaron porque no se podía elegir un tramo tóxico sino que toda la oración en sí lo es. Los datos para el entrenamiento del modelo

SPANS	TEXT
[12,13,14,15,16, 17,18,19,20,21]	Lol. What a ridiculous insult.
[55,56,57,58,59,60]	What if his opinion is that most other commenters are idiots?:-)
[80,81,82,83,84,85, 86,87,88,89,176,177, 178,179,180]	Survival of the fittest would not have produced you. You are alive because your weak blood is supported by welfare and food stamps. Please don't reference Darwin in your icon. Loser.
[]	So tired of all these Portlanders moving to California and ruining the culture. When will it stop?!?

Tabla 1: Formato de los datos utilizados.

vienen divididos en dos archivos a los que se les refiere como *train* y *trial*.

2.2 Análisis Exploratorio

El archivo *trial* contiene 690 textos, de los cuales 43 están marcados como que no se puede marcar tramos tóxicos como tal. Mientras que el archivo *train* contiene 7936 textos, de los cuales 485 se toman como que no se puede subrayar uno o varios tramos tóxicos como tal.

Al unir los dos documentos en uno sólo para tener un único documento de entrenamiento se observó que sólo habían 5 archivos repetidos entre ambos textos, se eliminan dichas repeticiones.

Es importante mencionar que en el texto se encontraron ciertos caracteres especiales con los que había que lidiar, por ejemplo, a lo que conocemos como emojis o símbolos que so de uso común en internet, como guiones, comillas, etc.

2.3 Preprocesamiento

Se mostraron unos ejemplos en la Tabla 1 de cómo es que venían idealmente los datos, sin embargo se tenían unas inconsistencias a la hora de analizar los datos reales, tales como que el subrayado estaba mal en cuanto a que se subrayaron palabras incompletas, se subrayaban espacios en ciertas ocasiones o también sucedía que se subrayaba casi todo el texto salvo signos de puntuación, lo que debía haberse subrayado como todo el texto, es decir, con

una lista vacía. Se realizaron procedimientos para corregir todos estos errores por cada oración. Se hizo además una limpieza en cuanto a caracteres que no eran de interés, dejando símbolos importantes del inglés como los signos de admiración, de interrogación, *ampersand*, guiones, signo de porcentaje, entre otros.

2.4 Modelos

Se utilizaron tres modelos para intentar resolver esta tarea: el Modelo Oculto de Markov, perceptrón de etiquetado y red neuronal tipo LSTM.

Para dos de los tres modelos utilizados (Modelo oculto de Markov y Perceptrón) fue necesario entrenarlos con listas de tuplas, donde cada lista contenía una oración que se descomponía por palabras y su etiquetado, estos dos elementos conformaban cada tupla. Se realizó un primer etiquetado sólo con palabras tóxicas y no tóxicas para estos dos modelos, luego se utilizó un sistema de etiquetado POS de la biblioteca de `nltk`(NLTK, 2020b) que el modelo pudiera interpretar de mejor manera cuando se tenían palabras tóxicas y qué tipo de etiquetado de POS tenía su contexto. Para la entrada del modelo LSTM sólo se utilizó el etiquetado tóxico y no tóxico pero como entrada recibe una lista con las palabras de la oración y además una lista con las etiquetas.

2.4.1 Modelo oculto de Markov

Para este modelo se utilizó una biblioteca propia de `nltk` (NLTK, 2020a) que para instanciar sólo necesita como argumentos las etiquetas que se estarán utilizando y el vocabulario utilizado. Las entradas son tal como se explican en la subsección anterior y nos regresa valores tipo como los de la entrada, a lo que sólo habría que hacerle el preprocesamiento inverso, convertir de palabras tóxicas a índices.

2.4.2 Perceptrón

Este modelo igualmente está incluido en `nltk`. A diferencia del modelo oculto de Markov, éste no necesita el vocabulario ni las etiquetas de antemano.

2.4.3 Red Neuronal

Para la implementación de la red neuronal se usó `PyTorch`.

Una representación visual del modelo se encuentra en la figura 5.

Se empieza con una capa de *embedding*, que en este caso lo que hace es recibir una palabra codificada como un entero y devolver el respectivo embedding. Al principio son aleatorios y con el entrenamiento van corrigiéndose.

Después, se tiene la capa protagonista de la red: una capa con 16 celdas LSTM. Finalmente, se pasa por una capa completamente conectada con dos neuronas (correspondientes a los tags tóxico o no tóxico) cuya función de activación fue log-softmax.

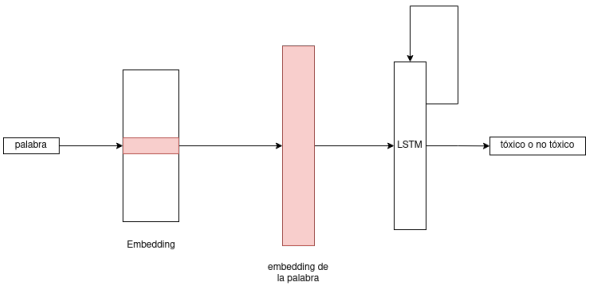

Figura 5: Topología de la red empleada

Para entrenarse, primero debía asignarse cada palabra de las oraciones a un entero, empezando por cero. Igualmente se debía codificar los tags (cero para tóxico, uno lo contrario). Se usó la log-verosimilitud negativa como función de pérdida (7), y descenso del gradiente estocástico como optimizador.

$$L = -\sum_{n=1}^{N} \frac{w_n x_{n,y_n}}{\sum_{n=1}^{N} w_n} \tag{7}$$

El código que se utilizó para todo lo explicado anteriormente se encuentra en (Óscar Alvarado & Dante Bermúdez & Martín García, 2021).

3 Resultados y análisis.

La evaluación de los modelos fue hecha mediante la métrica F1.

Sea S_A^t el conjunto de índices que corresponden a tópicos tóxicos que se obtiene al aplicar el sistema de detección A sobre un texto t, y sea S_G^t el conjunto de índices que fueron anotados por las personas.

Entonces, definimos la métrica F1 como

$$F_1^t(S_A^t, S_G^t) = 2\frac{P^t(S_A^t, S_G^t)R^t(S_A^t, S_G^t)}{P^t(S_A^t, S_G^t) + R^t(S_A^t, S_G^t)} \tag{8}$$

donde

$$P^t(S_A^t, S_G^t) = \frac{|S_A^t \cap S_G^t|}{|S_A^t|} \tag{9}$$

$$R^t(S_A^t, S_G^t) = \frac{|S_A^t \cap S_G^t|}{|S_G^t|} \tag{10}$$

A la expresión (9) se le conoce como precisión y a (10) como sensibilidad del modelo A, por lo que el F_1 no es mas que la media armónica entre estas dos.

Si S_G^t y S_A^t están vacíos, entonces $F_1 = 1$, pero si S_A^t no lo está, entonces $F_1 = 0$

En la tabla 2 se reportan la métrica descrita para cada uno de los modelos desarrollados con clasificación binaria.

	train	test
Modelo Oculto de Markov	0.585	0.284
Perceptron	0.933	0.490
Red neuronal	0.604	0.437

Tabla 2: Desempeño de modelos binarios usando F1.

Por otro lado, en la tabla 3 se reportan la métrica descrita para cada uno de los modelos desarrollados con el etiquetado gramatical.

	train	test
Modelo oculto de Markov	0.660	0.310
Perceptron	0.890	0.491

Tabla 3: Desempeño de modelos con variante POS usando F1.

Se observa que los mejores resultados obtenidos son con el modelo de perceptrón. Con el etiquetado POS se obtienen mejores resultados que sólo con el etiquetado tóxico y no tóxico, pero no son abruptamente mejores como sucedió usando el etiquetado POS con el modelo oculto de Markov.

4 Conclusiones

El perceptrón con el etiquetado POS presenta los mejores resultados, que se pueden mejorar etiquetando como tóxicos los espacios que hay entre las palabras tóxicas, ya que es un patrón que se observa en los subrayados. Como parte de esto, es importante indicar que, a pesar de que se contrató a los usuarios de "alta calidad" para que subrayaran el texto, el subrayado era bastante malo, lo que afectaba el rendimiento de cada modelo por más preprocesamiento que se hiciera ya que eran errores humanos que no se podrían corregir.

La métrica del modelo no refleja del todo el desempeño del mismo en una aplicación real.

Referencias

Óscar Alvarado & Dante Bermúdez & Martín García. 2021. Toxic-spans.

Matthew Honnibal. 2013. A good part-of-speech tagger in about 200 lines of python.

Daniel Jurafsky & James H. Martin. 2020. https://web.stanford.edu/ jurafsky/slp3/a.pdf.

NLTK. 2020a. Hmm tagger.

NLTK. 2020b. Pos tagger.

PyTorch. 2019. Lstm.