Toxic Spans Detection

Mario Garrido
IIMAS, UNAM
mg@czacki.com

David Guzmán
IIMAS, UNAM
david_guzmanr@ciencias.unam.mx

Alejandro Hernández
IIMAS, UNAM
roher1727@gmail.com

Abstract

La moderación del lenguaje tóxico en la red es un tema importante para promover discusiones sanas en línea, sin embargo, la mayoría de los modelos de detección de toxicidad no identifican los tramos que hacen que una publicación sea tóxica. En este trabajo creamos un modelo que es capaz de identificar tramos tóxicos en publicaciones en inglés, logrando un F1 score de 0.6488, el cual es un desempeño aceptable para la competencia (SemEval, 2021) en la que se participó.

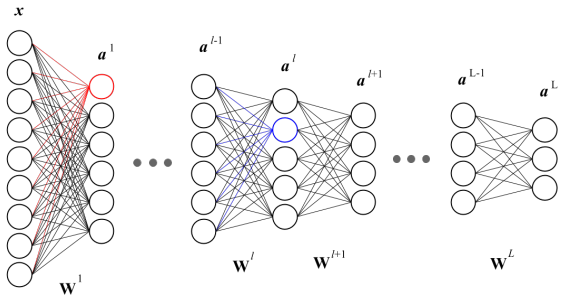

Figure 1: Representación gráfica de una red neuronal/perceptrón multicapa.

1. Introducción

A lo largo del siguiente trabajo se tratarán conceptos que no han sido elaborados a lo largo del curso. Debido a esto, procederemos a dar una breve introducción a los mismos.

1.1. Redes Neuronales Artificiales

Abordamos a las redes neuronales artificiales como nuestro método primario de solución debido a que tienen la capacidad de generar modelos de bajo sesgo y varianza si son entrenadas con grandes cantidades de datos. Una red neuronal artificial puede ser entendida como una composición de funciones $f_1(f_2(f_3(...f_n(\mathbf{x}|\theta))))$ en la cual se tiene una serie de parámetros θ estimados de un conjunto de entrenamiento en base a la arquitectura de funciones o capas utilizadas, y un conjunto de datos sobre el cual se desea generar una predicción. Debido a la complejidad de estos modelos, es posible utilizarlos sobre un conjunto de datos grande para resolver toda índole de problemas.

El método principal de predicción para una red neuronal artificial es mediante el uso de combinaciones lineales de las entradas de una "capa" y una función aplicada a esta misma. La función (denominada función de activación) permite introducir un comportamiento no lineal en el sistema, lo que a su vez genera la posibilidad de simplemente crecer la red para reducir el sesgo del modelo y de utilizar métodos de regularización para reducir la varianza del mismo. Una red neuronal bien entrenada es entonces capaz de generar funciones de predicción arbitrariamente complejas para ajustar a un modelo de datos.

1.2. GloVe embeddings

GloVe es un algoritmo no supervisado de aprendizaje que permite obtener la representación vectorial de un conjunto de palabras. El entrenamiento de estos embeddings se realiza en base a los estadísticos de coocurrencia de un corpus y esto genera una representación de las subestructuras lineales presentes en un corpus.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 2: Ilustración del funcionamiento de los embeddings GloVe. Utilizando dos palabras y su coocurencia con varias otras es posible generar una proporción de probabilidades que indica que tan relevante es la palabra respecto a otra. Esta es la base del modelo que permite entrenar a los embeddings utilizando una técnica de factorización de matrices aproximada.

Aunque no ahondaremos en la metodología matemática del modelo, el objetivo de entrenamiento del mismo es generar embeddings con la propiedad de que los vectores con el producto punto de dos representaciones de palabras sea aproximadamente igual al logaritmo de la probabilidad de coocurrencia del par de palabras. De esta forma se generan vectores con propiedades propicias para generar analogías como las de Word2Vec.

1.3. Red neuronal recurrente

Una red neuronal recurrente (o RNN) es una red neuronal diseñada para lidiar con secuencias de datos. Esta funciona como la generalización de una red neuronal tradicional al implementar un sistema de memoria de salidas pasadas de la red. En esta, básicamente se toma como una entrada a la salida pasada de la misma red. Esto genera un "vector de estados pasados" que permite a la red tomar en cuenta los elementos previos en una secuencia para generar una predicción sobre estos.

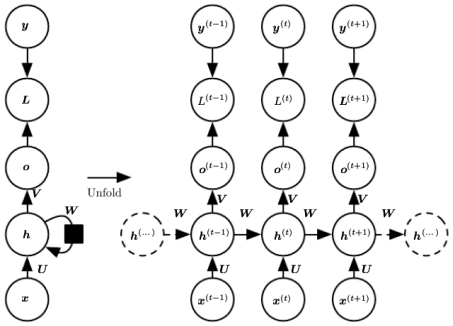

Figure 3: Ilustración del funcionamiento de una red neuronal recurrente. Nótese como a parte de la entrada en cada punto de la secuencia, se tiene el estado oculto de la misma red en la unidad de tiempo anterior.

Existen algunos problemas con la formulación tradicional de las redes neuronales recurrentes, principalmente en el hecho de que les afecta fuertemente el problema del desvanecimiento de gradiente, y que tienen "memoria a corto plazo" debido a que el vector de estados pasados es un vector con perdidas de información, por lo que se ve dominado por la información del pasado inmediato presenciado por la red.

1.4. LSTM

Una long short-term memory neural network es una red neuronal diseñada para lidiar con secuencias de datos que parte del diseño de una red neuronal recurrente. A cada capa de una red LSTM se le agregan "compuertas" que permiten a la red regular que tanto recuerda y olvida de cada elemento de la secuencia. Esto permite al mismo sistema recordar la información más relevante en una secuencia.

1.5. LSTM Bidireccional

Para este trabajo se utilizó una LSTM bidireccional, que permite tomar en cuenta el pasado y futuro de la secuencia en cada punto de la misma. Es virtualmente idéntica a las LSTMs tradicionales, solo agregando el vector de salida tomando en cuenta la secuencia en orden de futuro a presente. Debido a que la tarea a abordar en este proyecto era altamente dependiente del contexto, se optó por esta arquitectura para tener un mejor desempeño en el etiquetado de tokens individuales.

2. Problemática

La moderación es crucial para promover discusiones sanas en línea. Aunque se ha publicado varios conjuntos de datos y modelos de detección de toxicidad (el cual en este trabajo se refiere como lenguaje abusivo) la mayoría de ellos clasifican comentarios o documentos completos y no identifican los tramos que hacen que un texto sea tóxico. Sin embargo, resaltar estos tramos tóxicos puede ayudar a los moderadores humanos a identificar las secciones tóxicas en vez de sólo obtener un *puntuaje de toxicidad* generado por un sistema. La evaluación de sistemas que podrían localizar con precisión tramos tóxicos dentro de un texto es, por lo tanto, un paso crucial hacia una moderación semiautomatizada exitosa.

3. Descripción del corpus

Los datos se encuentran en (SemEval, 2021), son alrededor de 10 mil publicaciones que provienen del Civil Comments dataset y que fueron anotados por personas con las siguientes instrucciones: *Extraiga las secuencias de palabras tóxicas (tramos)*

del comentario, resaltando cada uno de esos tramos y luego haciendo clic en el botón derecho. Si el comentario no es tóxico o si se debe anotar todo el comentario, marque la casilla correspondiente y no resalte ningún intervalo.

Sin embargo, en algunas publicaciones tóxicas el mensaje central que se transmite puede ser inherentemente tóxico (por ejemplo, una publicación sarcástica que afirma indirectamente que las personas de un origen en particular son inferiores) y, por lo tanto, puede ser difícil atribuir la toxicidad de esas publicaciones a tramos particulares, por ejemplo, en la tercer y cuarta publicación de la tabla 1 no se etiquetaron tramos tóxicos ya que se decidió que toda la oración era tóxica.

Tramo	Texto
[4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17]	How fucking stupid are you?
[20, 21, 22, 23]	You are obviously a fool
[ ]	Dumb shit left shit in toilet, now he's up shit creek.
[ ]	This is a garbage country with garbage people.

Table 1: Ejemplos de anotaciones en el conjunto de entrenamiento.

Algunas estadísticas básicas del conjunto de datos se encuentran en la tabla 2, en el conjunto de entrenamiento un 6.11 % de las publicaciones fueron anotados con [ ], ya sea porque eran completamente tóxicos, contenían sarcasmo, o simplemente no eran tóxicos. Estas publicaciones resultan un problema ya que es posible que bajen el rendimiento de nuestro modelo, por lo que se intentaron abordar desde distintos ángulos.

Dataset	Tóxico	Vacío	Total
Train	7454	485	7939
Test	647	43	690
Evaluation	-	-	2000

Table 2: Estadísticas del conjunto de datos.

4. Metodología

A continuación se describe brevemente la metodología utilizada. El proceso seguido inició con la generación de los datos de entrenamiento (preprocesamiento), seguido de una etapa de entrenamiento del modelo y finalmente un posprocesamiento para mejorar el desempeño del modelo y ajustar las salidas del modelo a las esperadas por el problema.

4.1. Preprocesamiento

En algunas oraciones se ponen los espacios, comas y otros signos de puntuación como caracteres tóxicos, lo que en realidad no tiene mucho sentido y se decidió quitar la mayoría de los signos de puntuación ([',"\()+-.:;¡=¿[]^`{}|~] y espacios en blanco) que estaban marcados como tóxicos. Sin embargo, hay signos de puntuación que en efecto sí tienen una naturaleza tóxica cuando aparecen, por ejemplo, el signo * aparece frecuentemente cuando se trata de "censurar" una mala palabra, por lo que se decidió mantener otros signos de puntuación relevantes a la tarea.

Posteriormente se trató el tema de las publicaciones que estaban etiquetadas como [ ], los cuales se abordaron de tres diferentes maneras:

1. Se dejaron tal y como estaban.
2. Se etiquetaron como completamente tóxicos, es decir, toda la oración era tóxica.
3. Se removieron del conjunto de entrenamiento.

Con cada uno de los tres diferentes preprocesamientos para las publicaciones etiquetadas con [ ] se entrenó el modelo para posteriormente elegir el que mejor se desempeñaba en el conjunto de prueba. Por último, se pasó a minúsculas el conjunto de entrenamiento y se tokenizó con ayuda de spaCy.

4.2. Modelo de aprendizaje

Se generó una LSTM bidireccional con la siguiente arquitectura:

Primero se utiliza un conjunto de embeddings de GloVe preentrenados en un corpus originado en Twitter para generar vectores representativos de cada uno de nuestros tokens. Posteriormente, se ingresa la secuencia de embeddings a una LSTM bidireccional con las siguientes características:

- 6 capas internas en la sección LSTM.
- Vectores de salida de 600 elementos para cada uno de los sentidos de la LSTM (en total 1200 elementos).

- Probabilidad de dropout de 20 % en la LSTM para evitar el sobreajuste.
- Una capa densa de salida con una función de activación sigmoide para realizar una predicción binaria (no tóxico o tóxico).

Se realizó el entrenamiento con una función de pérdida de entropía cruzada binaria y un optimizador Adam. Aunque se utilizó una función de pérdida distinta a la que se evaluaría en la competencia, se observó que la mejora en una era aproximadamente equivalente a la otra. Debido a esto, y para intentar no sobreajustar al conjunto de entrenamiento, se siguió con este proceso hasta que la F1 empezó a decaer, tras lo cual seleccionamos los pesos de la red con el mejor desempeño en esta métrica como nuestro modelo final.

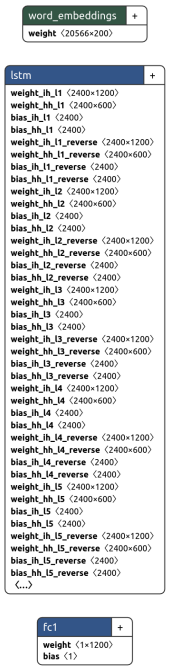

Figure 4: Arquitectura del modelo.

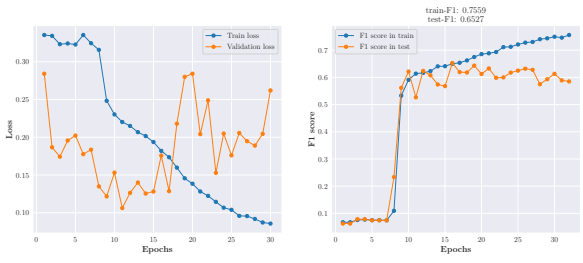

Figure 5: Gráfica de funciones de perdida y F1 score utilizadas en el entrenamiento. Debe observarse que se logró el mejor desempeño en la métrica F1 de evaluación en la época 16 de entrenamiento, por lo que esa fue la utilizada para el modelo final.

4.3. Postprocesamiento

Cabe aclarar que debido a que utilizamos un enfoque basado en tokenizar y el problema se evalúa en torno a la generación de índices, se transformaron las predicciones de la red a los índices propios del token en la oración original y se incluyeron como tóxicos todas las secuencias con longitud menor a ocho que estuvieran rodeadas de índices tóxicos de tal manera que $[1, 9] \rightarrow [1, 2, 3, 4, 5, 6, 7, 8, 9]$ pero $[1, 10] \rightarrow [1, 10]$. Esto permite cubrir algunas de las deficiencias en la salida del modelo como lo son espacios identificados como tóxicos (que no existen en la entrada por la tokenización) y a su vez permitiendo identificar palabras conectoras como tóxicas (tal y como está en el corpus original).

4.4. Intentos previos

Debido a que esta no es nuestra primera aproximación al problema, debemos hacer notar que antes de intentar con esta arquitectura habíamos utilizado modelos de Markov ocultos y campos aleatorios condicionales, sin embargo el desempeño obtenido era inferior al que logramos con las LSTMs, por lo que rápidamente abandonamos esta aproximación y nos enfocamos solamente a generar el modelo previamente descrito.

5. Resultados

Para evaluar las respuestas del modelo se usa el F_1 score. Sea entonces un modelo A_i que regresa un conjunto de caracteres $S^t_{A_i}$ que clasificó como tóxicos, sean G^t los caracteres tóxicos del *ground truth* del texto t, el F_1 score del modelo A_i con respecto al *ground truth* G para el texto t se define como

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)}, \quad (1)$$

donde

$$P^t(A_i, G) = \frac{|S^t_{A_i} \cap S^t_{A_i}|}{|S^t_{A_i}|}$$

$$R^t(A_i, G) = \frac{|S^t_{A_i} \cap S^t_{A_i}|}{|S^t_G|}$$

Si S^t_G es vacío para algún texto t entonces $F_1^t(A_i, G) = 1$ si $S^t_{A_i}$ también es vacío y $F_1^t(A_i, G) = 0$ en otro caso. Finalmente, se promedia $F_1^t(A_i, G)$ sobre todos los textos t de un

conjunto de datos de evaluación para obtener un solo puntaje para el sistema A_i.

En la tabla 3 se muestran los resultados de nuestro modelo al aplicar distintos preprocesamientos al conjunto de datos de entrenamiento, se obtuvo que poner las publicaciones con [ ] como completamente tóxicas ayudaba al modelo a generalizar mejor en el conjunto de prueba, probablemente debido a que *sesgábamos* el modelo a poner más palabras como tóxicas.

Preprocesamiento	F1 train	F1 test
Dejar las publicaciones con [ ] tal como están	0.6198	0.6258
Poner las publicaciones con [ ] como completamente tóxicos	0.6498	0.6526
Remover las publicaciones con [ ] del conjunto de entrenamiento	0.7260	0.6459

Table 3: Resultados con distintos preprocesamientos.

Particularmente, logramos obtener un score de 0.6488 en el conjunto de datos de evaluación en (SemEval, 2021), siendo el máximo score de los otros participantes hasta el momento de 0.7, por lo que nuestro modelo parece hacer un buen trabajo considerando la mala calidad del etiquetado en los datos.

6. Conclusiones

Creemos que dados los resultados de otros participantes en Codalab, nuestro modelo tuvo un desempeño aceptable para la tarea establecida. Dado que la máxima calificación obtenida parece ser un 0.7 en F1 score y nuestro modelo presentó un 0.6488 en la misma métrica, creemos que tenemos un modelo funcional (aunque no del estado del arte). Si el tiempo y los recursos lo hubieran permitido, teníamos algunas ideas para mejorar el desempeño más:

- Generar un conjunto de datos aumentados para entrenar la red con más datos.
- Utilizar un modelo de secuencia basado en caracteres en vez de tokens.
- Implementar la generación de características mediante convolución de los caracteres en el token para ampliar el poder predictivo de los embeddings.
- Experimentar con un modelo basado en atención.

No obstante de esto, el modelo parece funcionar para la tarea en cuestión y está mejor evaluado que el de muchos otros participantes, por lo que estamos satisfechos con el trabajo realizado.

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation.

SemEval. 2021. Toxic spans detection.