Don't Patronize Me!

Marcos Yañez Espindola
UNAM / IIMAS
marcos080399@hotmail.com

David Vázquez Rojas
UNAM / IIMAS
david_vrj@outlook.com

Luis Fernando Flores Tiburcio
UNAM / IIMAS
fer.f.tiburcio@gmail.com

Abstract

Los retos que se presentan para los algoritmos del entendimiento lingüistico no permiten un formalismo en la obtención de resultados esperados. Uno de ellos claramente es el identificar si en una oración contiene lenguaje condescendiente o no, para ello existen diversos algoritmos que nos permiten aproximar un modelo adecuado para lograr un clasificador decente. Lo anterior involucra diversos factores sujeto a la lengua que se observa a lo largo de la implementación.

1 Introducción

El Lenguaje Condescendiente y Despectivo (PCL, por sus siglas en inglés) es a menudo involuntario e inconsciente. Por otro lado, debido a su sutileza, subjetividad y las (generalmente) buenas intenciones detrás de su uso, la audiencia a menudo desconoce este trato decreciente. Pero el PCL puede ser potencialmente muy dañino, ya que alimenta estereotipos, rutiniza la discriminación y conduce a una mayor exclusión.

Alguien es Condescendiente y Despectivo cuando su lenguaje denota una actitud superior hacia los demás, les habla mal o los describe a ellos oa su situación de una manera caritativa, lo que genera un sentimiento de lástima y compasión.

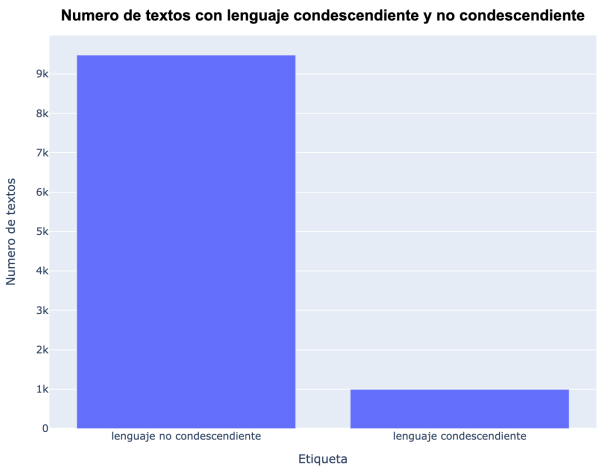
La detección de PCL es difícil tanto para los humanos como para los sistemas de PNL, debido a su naturaleza sutil, su subjetividad y la gran cantidad de conocimiento del mundo y razonamiento de sentido común que se requiere para comprender este tipo de lenguaje. Con esta tarea, esperamos ampliar los límites de este nuevo desafío en la comunidad de PNL.

El objetivo del proyecto es hacer clasificación binaria, es decir dado un párrafo decir si este contiene o no PCL y obtener el mejor F1 score posible, para esto, probaremos diferentes características tales como bolsa de palabras, tf-idf, bigramas, trigramas y word embeddings y nos quedaremos con la que nos de un mejor F1 score.

2 Metodología

2.1 Obtencion de los datos

Para el desarrollo de esta tarea, los organizadores del concurso nos proporcionaron un corpus etiquetado el cual consta de 10469 párrafos, de los cuales 9476 no contienen lenguaje condescendiente y tan solo 993 contienen lenguaje condescendiente, en un inicio esto ya representa un problema debido a que al haber muchos mas párrafos sin lenguaje condescendiente, nuestros modelos se sesgaran hacia esta clase.

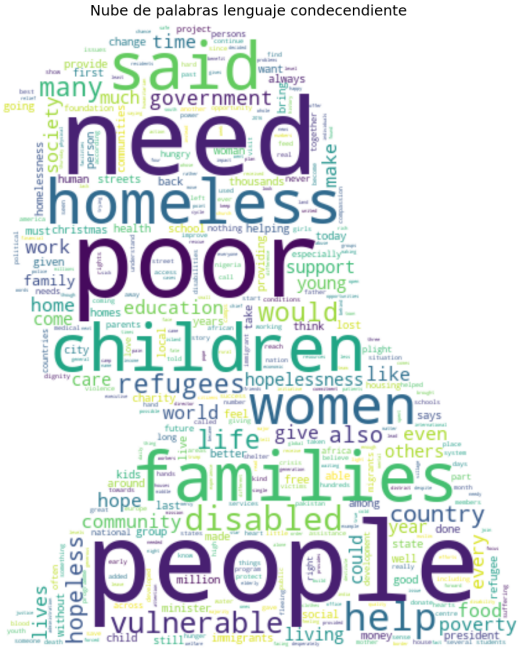


2.2 Analisis Exploratorio

Antes de empezar con el modelado, se decidió hacer exploración de los datos, para ello obtuvimos las palabras, bigramas y trigramas mas frecuentes para ambas clases, con el objetivo de ver si hay una diferencia significativa y los resultados son los siguientes:

Palabras

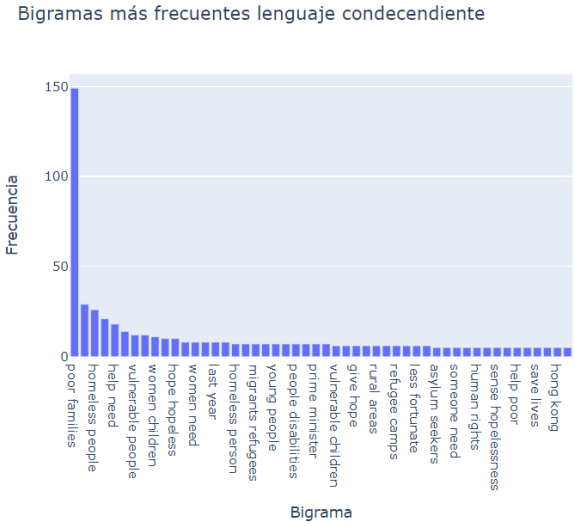
Nube de palabras lenguaje condecendiente



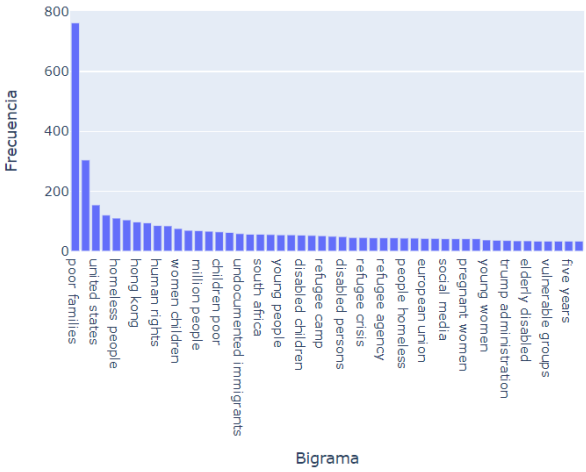
Nube de palabras lenguaje no condecendiente

Bigramas

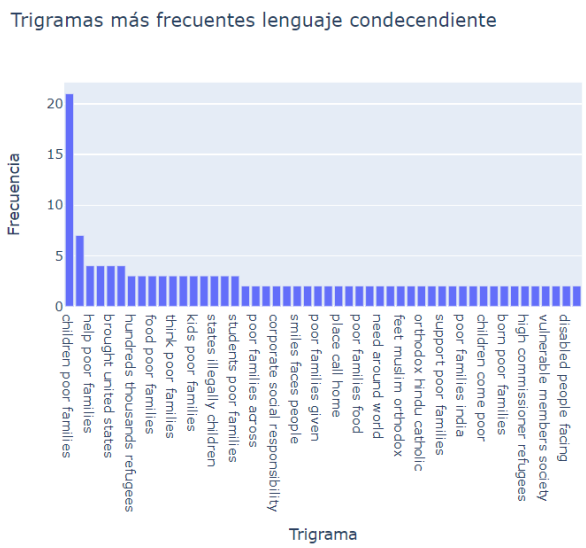
Bigramas más frecuentes lenguaje condecendiente

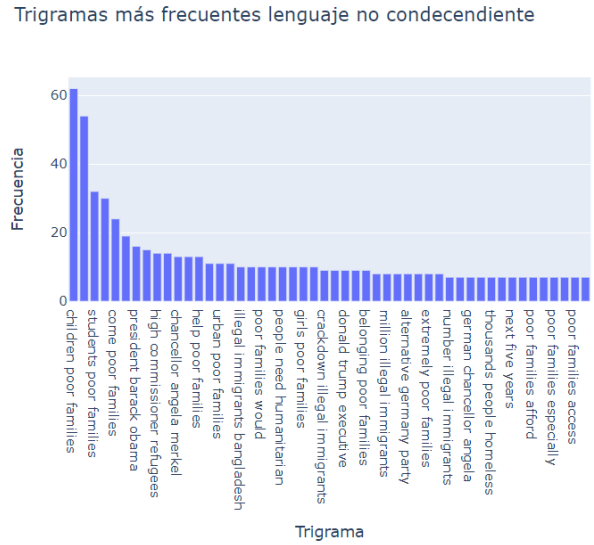


Bigramas más frecuentes lenguaje no condecendiente



Trigramas





Observamos que no existe una diferencia muy clara entre las palabras, bigramas y trigramas mas frecuentes, esto lo ascociamos con en la practica, no es muy importante las palabras que usas para ser condescendiente, si no como las usas, es por eso que esta tarea se complicara bastante.

2.3 Limpieza y Preprocesamiento

Antes de empezar con el modelado, para cada uno de los parrafos se hizo lo siguiente:

- Se eliminaron todos los caracteres no alfanuméricos
- Se remplazaron todos los saltos de linea, tabulaciones y espacios dobles por un solo espacio
- Se pasaron a minúscula
- se eliminaron las stopwords

Además de esto, notamos algunos problemas imposibles de corregir automáticamente, como errores de ortografía y espacios en medio de las palabras (por ejemplo Ju arez cuando deberia de ser Juarez), los cuales eran bastantes y pensamos que impactaran de forma negativa en los modelos, sin embargo debido al tiempo no se pudieron corregir manualmente.

2.4 Modelado

Para la parte del modelado y las pruebas usamos una red neuronal muy simple, la cual esta formada por una capa de entrada, una capa de salida y una capa intermedia. Para la capa de entrada tenemos tantas neuronas como características (palabras, bigramas, trigramas) y tiene función de activación relu, luego la capa intermedia siempre tiene 10 neuronas (el numero 10 se decidió empíricamente) con activación sigmoide y finalmente la capa de salida es una sola neurona. Todas las capas anteriores son densas, es decir cada neurona de una capa esta conectada con todas las neuronas de la capa siguiente. Este modelo se implemento usando keras, debido a la facilidad para programar redes neuronales.

Una vez tenemos este modelo, probaremos usando:

- Bolsa de palabras simple (20,000 palabras)
- Bolsa de palabras TF-IDF (20,000 palabras)
- Palabras y Bigramas (tamaño del vocabulario: 100,000)
- Palabras, Bigramas y Trigramas (tamaño del vocabulario: 200,000)
- Word Embeddings (20,000 palabras) de tamano 300

Hardware y tiempo de entrenamiento

Para el entrenamiento, usamos el optimizador Adam con learning rate de 0.001 y 50 epocas con un batch size de 10. El hardware empleado fue un i7 7700k con una RTX 2060 super y tomo aproximadamente 15 min por modelo (1:15 hora aprox).

Conjuntos de entrenamiento, pruebas y validación

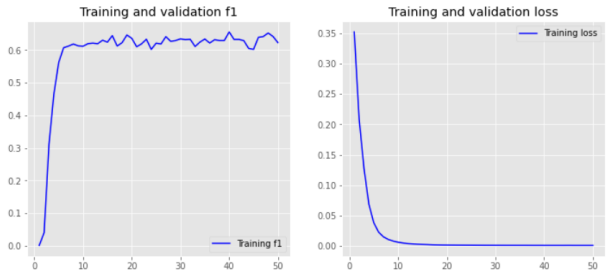
De los datos usamos el 25% como validación, y del 75% restante tomamos un 10% como conjunto

de pruebas, el resto lo usamos como entrenamiento para cada uno de los modelos. Al final tomamos el que de un mejor F1 score en el conjunto de pruebas y usamos ese modelo para hacer predicciones en el conjunto de validación y asi obtener el F1 score que reportamos.

Para poder replicar los resultados usamos la semilla: 68710241

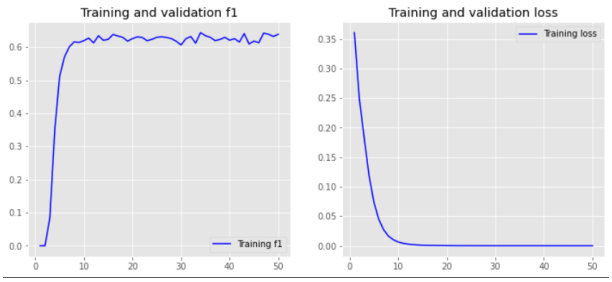
3 Resultados

Bolsa de palabras simple



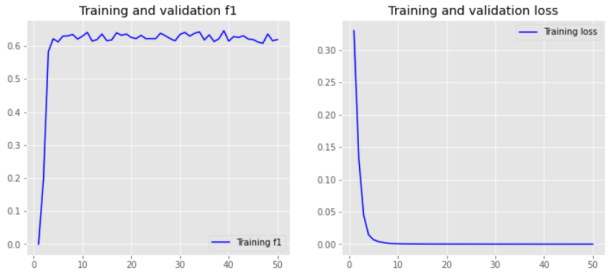
Para este modelo obtuvimos un F1-score de 0.2730 en el conjunto de pruebas

Bolsa de palabras TF-IDF



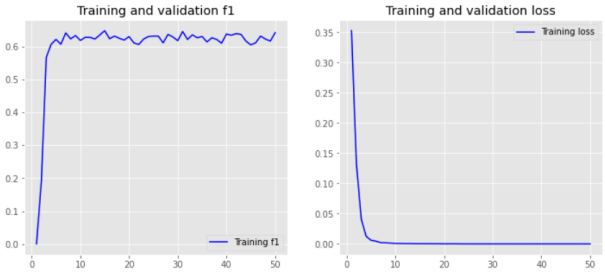
Para este modelo obtuvimos un F1-score de 0.2750 en el conjunto de pruebas

Bigramas



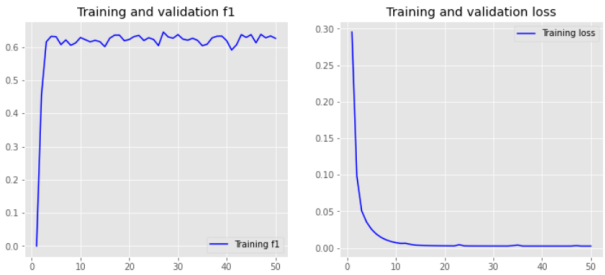
Para este modelo obtuvimos un F1-score de 0.2494 en el conjunto de pruebas

Trigramas



Para este modelo obtuvimos un F1-score de 0.2221 en el conjunto de pruebas

Embeddings



Para este modelo obtuvimos un F1-score de 0.2434 en el conjunto de pruebas

4 Analisis de los Resultados

Observamos que a todos los modelos les toma menos de 10 épocas para llegar a una cota en el F1 y la perdida, esto debido a que hasta ese punto llega la separabilidad de las clases.

También observamos que el mejor modelo considerando el F1-score en el conjunto de prueba es el que usa bolsa de palabras con TF-IDF, lo cual es curioso, ya que nosotros teníamos bastantes expectativas en los modelos con ngramas y el de los word embeddings, sin embargo para los modelos de ngramas observamos que entre mas ngramas usamos, menos era el F1-score, además la memoria y tiempo de procesamiento aumentaba demasiado, siendo que para los trigramas, el numero de parámetros a optimizar ya era de 2,000,021.

Por lo tanto el modelo que probamos con el conjunto de validación sera el basado en TF-IDF y obtuvimos el siguiente resultado:

```
task1_precision:0.25667351129363447
task1_recall:0.49407114624505927
task1_f1:0.33783783783783783
```

Como podemos observar un F1 score de 0.33783 es un resultado demasiado bajo, sin embargo por la dificultad que representa esta tarea, lo consideramos adecuado. Nosotros asociamos esto con los problemas antes mencionados, siendo el primero que el corpus esta demasiado desbalanceado, ademas contiene demasiadas faltas de ortografía y errores, y por ultimo también es importante mencionar que el poder computacional fue una gran limitante, ya que no podemos tomar un vocabulario tan extenso como quisiéramos.

5 Conclusiones

Como ya mencionamos, la clasificacion de textos no una tarea sencilla, sobre todo cuando la diferencia entre las clases es demasiado sutil como en este caso, ya que incluso para los humanos nos es difícil identificar el lenguaje condescendiente, por ello muchas veces lo usamos inconscientemente, sin embargo, considerando que usamos un modelo muy sencillo de aprendizaje de maquina y todos los problemas antes mencionados, el resultado obtenido fue bastante bueno, aunque muy mejorable. Tambien es importante destacar que a diferencia de lo que esperabamos, el uso de ngramas y word embeddings no mejora el F1 score para esta tarea y usando este modelo en especifico. Como trabajo futuro, quedaria probar con algun modelo de red neuronal mas complejo y haciendo un procesamiento de datos mas complejo como usar steming o lematización, las cuales decidimos no usar en este proyecto ya que nuestro objetivo era comparar si en verdad el uso de ngramas y word embeddings eran de utilidad para esta tarea.

References

Basant Agarwal and Namita Mittal. 2014. Text classification using machine learning methods-a survey. In *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012*, pages 701–709. Springer.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309.

Carla Perez Almendros, Luis Espinosa Anke, and Steven Schockaert. 2020. Don't patronize me! an annotated dataset with patronizing and condescending language towards vulnerable communities. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5891–5902, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Real Python. 2021. Practical text classification with python and keras.

Keras Team. Keras documentation: Keras api reference.