Dime qué escribiste y te diré quién eres:

Análisis de sentimiento a AMLO en Twitter

Miguel Lerdo de Tejada
Eduardo Zago
Álvaro Morales
Sergio Parra
Siha Silva
México Vergara

Problema:

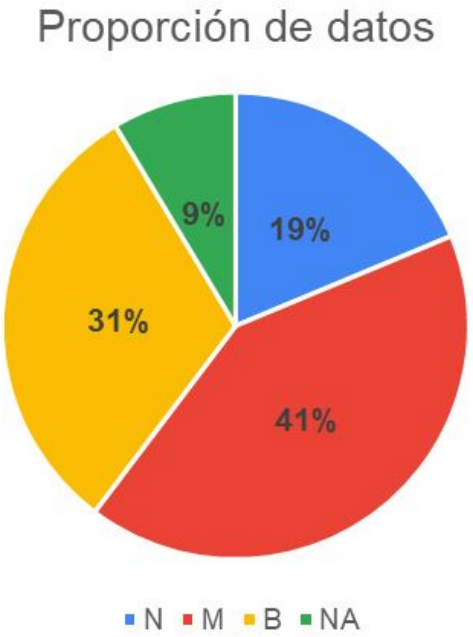
- La aprobación o desaprobación general de políticas o acciones gubernamentales es una variable importante en una democracia.
- Sin embargo, **los métodos de recolección de opiniones en ocasiones presentan distintas dificultades y fallas** técnicas que impiden que se tenga este indicador de manera rápida y veraz.
- Distintos problemas que se encuentran en la recolección de estos datos son:
    - Falta de recursos para realizar encuestas
    - Distintos tipos de sesgos al realizarlas (selección, aceptación, entre otros)

Alternativa: Usar modelos de DL para estimar una dimensión de la aprobación pública.

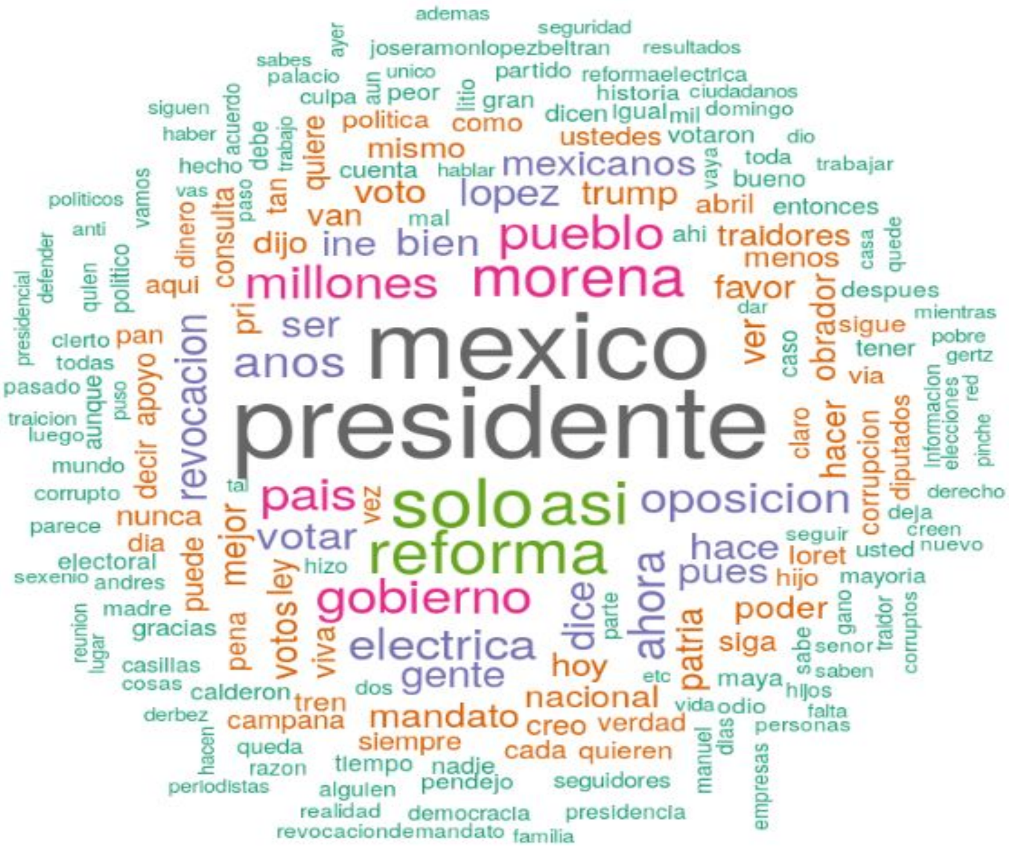
Datos

- Base de datos con 5600 registros y 8 variables extraídos del API de Twitter y clasificados equitativamente por el equipo para hacer un modelo de aprendizaje supervisado.

- Cada tweet se categorizó como:
  - Bueno/a favor de AMLO = B => 1740 datos
  - Malo/en contra de AMLO = M => 2330 datos
  - Neutral= N => 1048 datos
  - No de AMLO = NA => 482 datos (outliers)



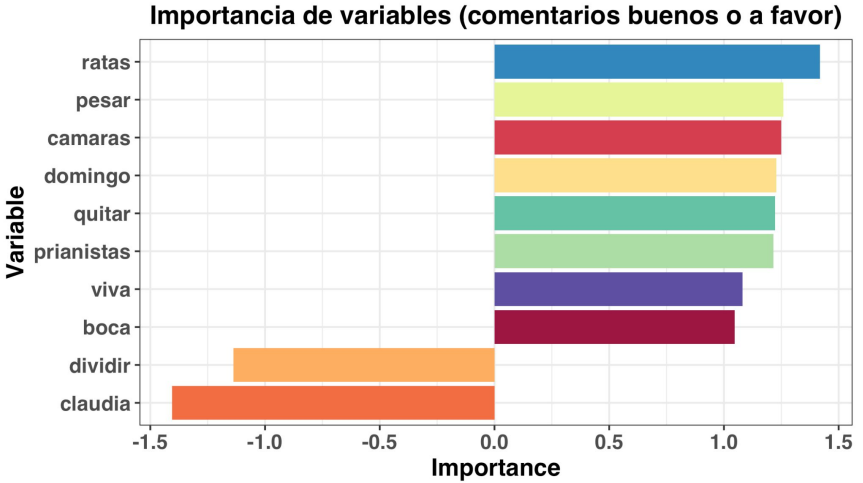
Para realizar el análisis de sentimiento se usaron dos modelos de Deep Learning (DL) y uno de Machine Learning (ML).

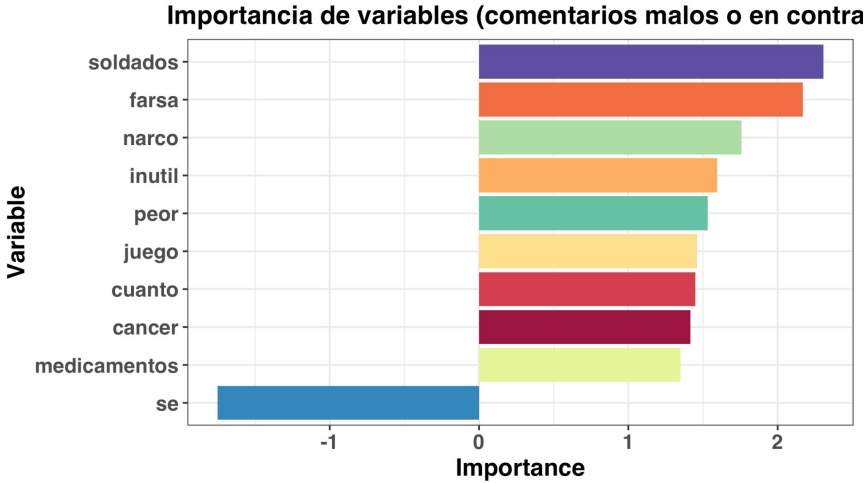


Multinomial Lasso

Con la intención de comparar el desempeño de métodos tradicionales de ML contra métodos de DL, se hizo la estimación del siguiente Multinomial Lasso:

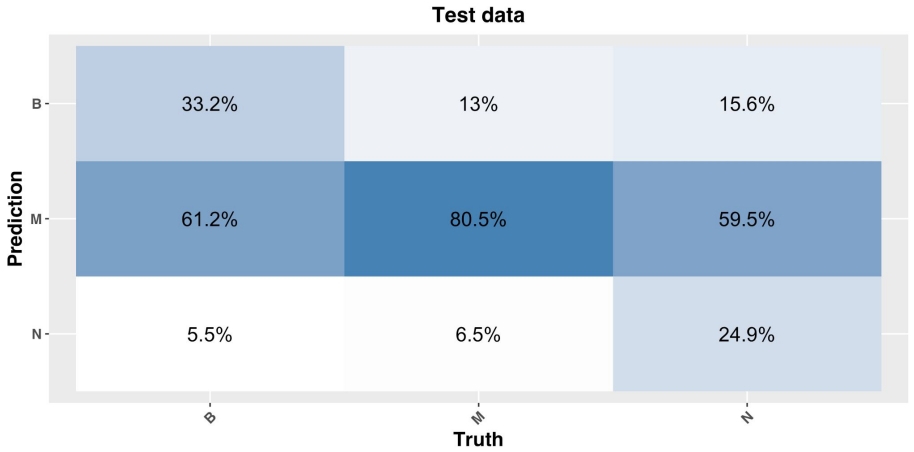
- Máximo número de variables (palabras) = 1,000.
- CV Folds = 10.
- Proporción de datos de entrenamiento = 80%.
- Se utilizó un rango de 30 lambdas de penalización.
- Tiempo de entrenamiento = 10 min. max
- 10 variables más importantes para definir clasificación:





Multinomial Lasso

- Comentarios “buenos” los clasifica mal debido a que los confunde con malos (groserías pueden ser un factor que incide en esta confusión).
- Comentarios neutros tiene una clasificación un poco arriba del azar (desempeño pobre).
- Accuracy = 52%.



BETO (pre-entrenado)

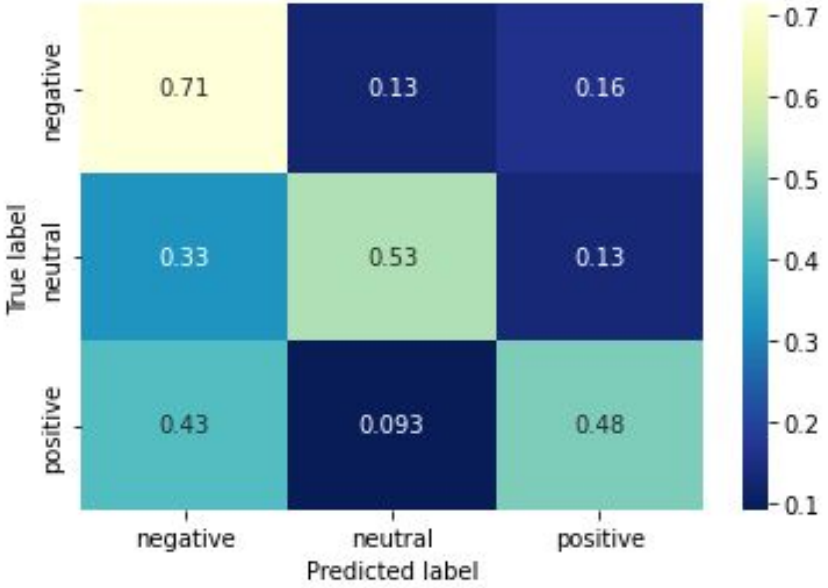
- En internet encontramos un BETO (BERT en español) ya construido y pre-entrenado que utilizamos para realizar nuestro análisis.

- Inicio (1er capa): Embeddings
- El BETO tiene 11 capaz (BERT- Hidden Layers), cada una tenía:
  - Capa de Atención (3 lineales con 768 “hidden units”, Dropout(p=0.1))
  - Capa “SelfOutput” (1 lineal, LayerNormalization y Dropout(p=0.1))
  - Capa Intermedia (1 lineal y activación GELU())
  - Capa Output (misma que “SelfOutput”)
- Pooling (capa final): (1 lineal y activación (classifier) Tanh())

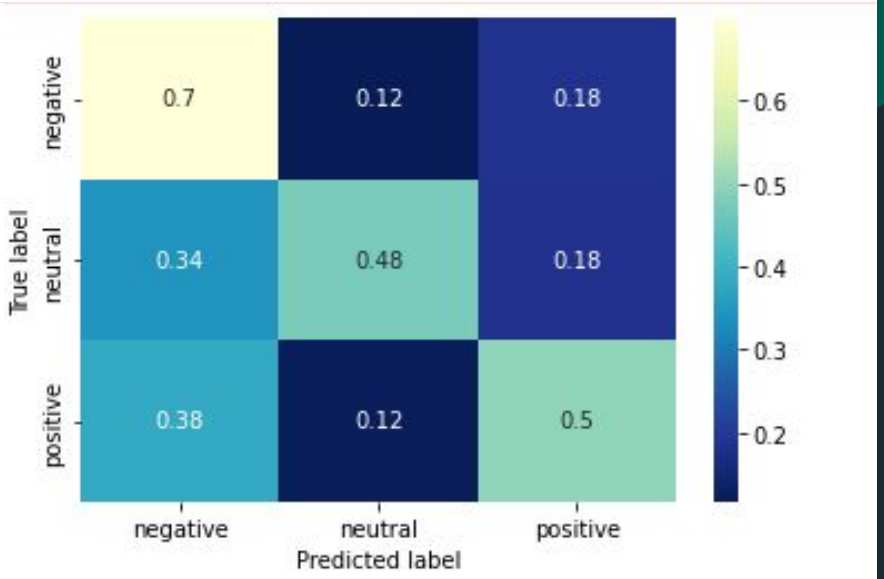
- Primero corrimos el modelo sin realizar fine-tuning, modificando solo hiper-parámetros como la función de optimización, el número máximo de tokens y modificando el pre-procesamiento del texto.
- La siguiente tabla muestra un estilo time-line de lo que modificamos y su desempeño (todos en una computadora de Google Cloud con 4vCPUs 15GB RAM)

Modelo	**Clases**	**Optimizador**	**Regularización**	**Pre-Procesamiento**	**Epochs/Time**	**Val Acc.**
BETO 1	3	AdamW	NO	max_tokens = 100	10/30 mins p/e	60%
BETO FT (Softmax)	3	AdamW	NO	max_tokens = 50	7/12 mins p/e	61%
BETO FT (Softmax) 2	3	SGD	NO	max_tokens = 50	7/15 mins p/e	59%
BETO 2	2	Adam	NO	max_tokens = 40, char > 1	8/8 mins p/e	67%
BETO 3	2	Adam	SI: l2 reg en la función de opt	max_tokens = 40, char > 1	3/ 20 mins p/e	60%
BETO 4	2	Adam	NO	max_tokens = 40	10/ 12 mins p/e	68%
BETO 5	2	Adam	SI: l2 reg en la función de opt	max_tokens = 40	10/ 12 mins p/e	68%

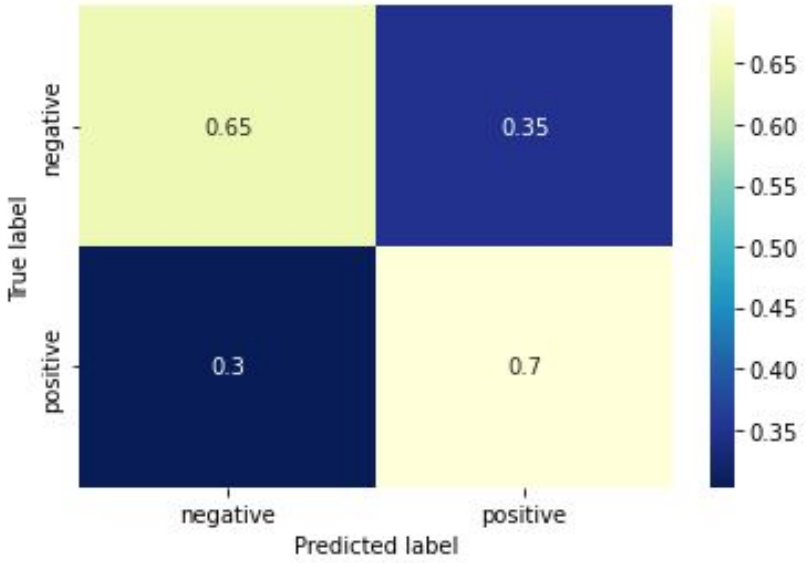
BETO 1


negative
neutral
positive
True label
0.71
0.13
0.16
0.33
0.53
0.13
0.43
0.093
0.48
negative
neutral
positive
Predicted label
0.7
0.6
0.5
0.4
0.3
0.2
0.1

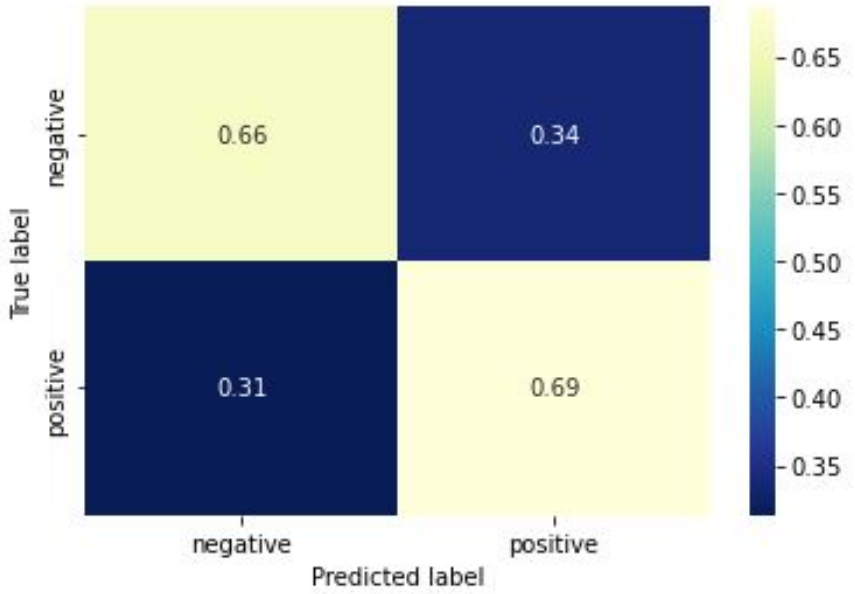
BETO 2

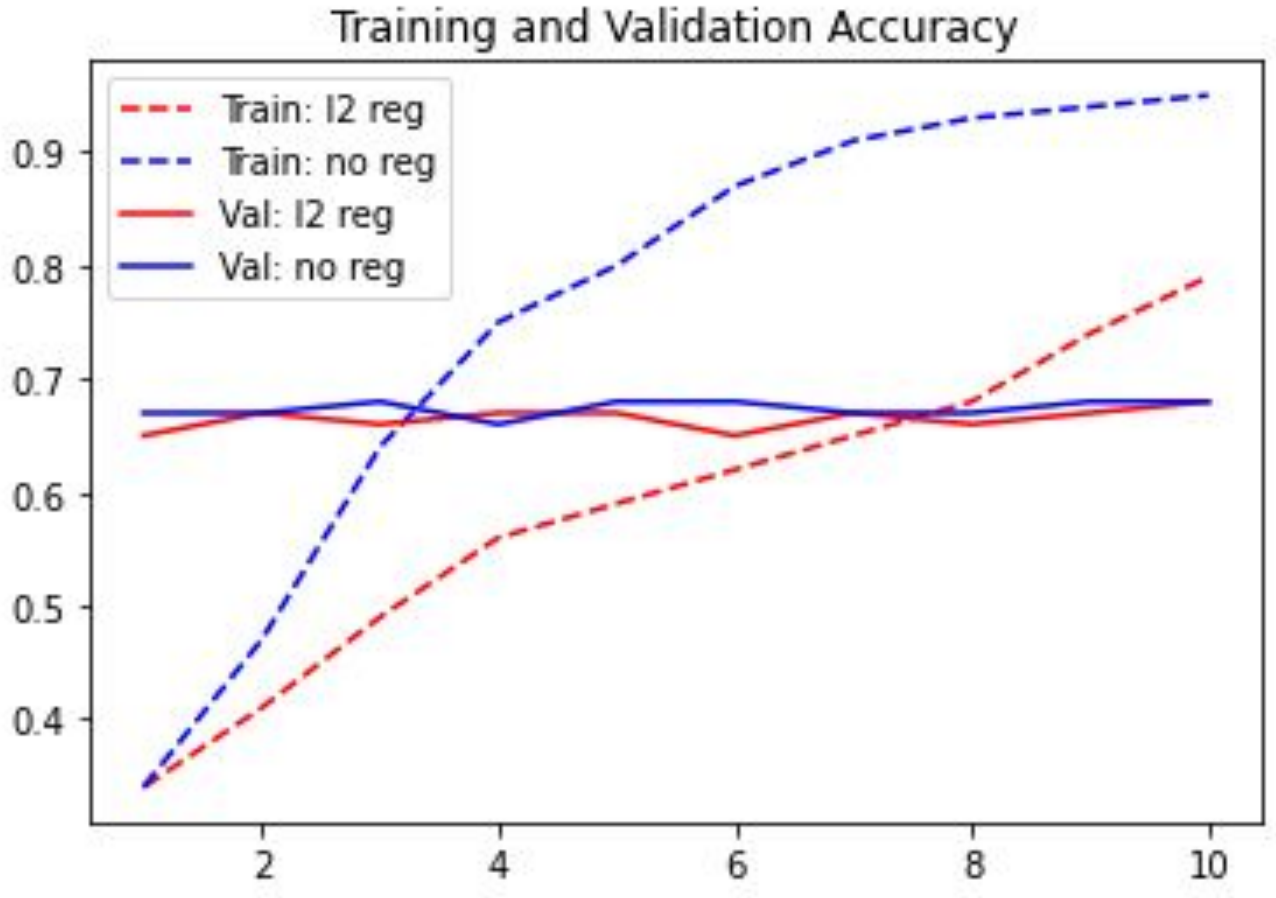

negative
neutral
positive
True label
0.7
0.12
0.18
0.34
0.48
0.18
0.38
0.12
0.5
negative
neutral
positive
Predicted label
0.6
0.5
0.4
0.3
0.2

BETO 4


0.65
0.35
0.3
0.7
negative
positive
True label
negative
positive
Predicted label
0.65
0.60
0.55
0.50
0.45
0.40
0.35

BETO 5


0.66
0.34
0.31
0.69
negative
positive
True label
negative
positive
Predicted label
0.65
0.60
0.55
0.50
0.45
0.40
0.35


Training and Validation Accuracy
Train: l2 reg
Train: no reg
Val: l2 reg
Val: no reg
0.9
0.8
0.7
0.6
0.5
0.4
2
4
6
8
10

LSTM

Otro modelo con potencial para lograr una precisión satisfactoria en la tarea de análisis de sentimiento es el LSTM bidireccional. Este modelo recursivo analiza las oraciones input de izquierda a derecha y de derecha a izquierda. Las pérdidas y precisiones asociadas son las siguientes:

Training set:

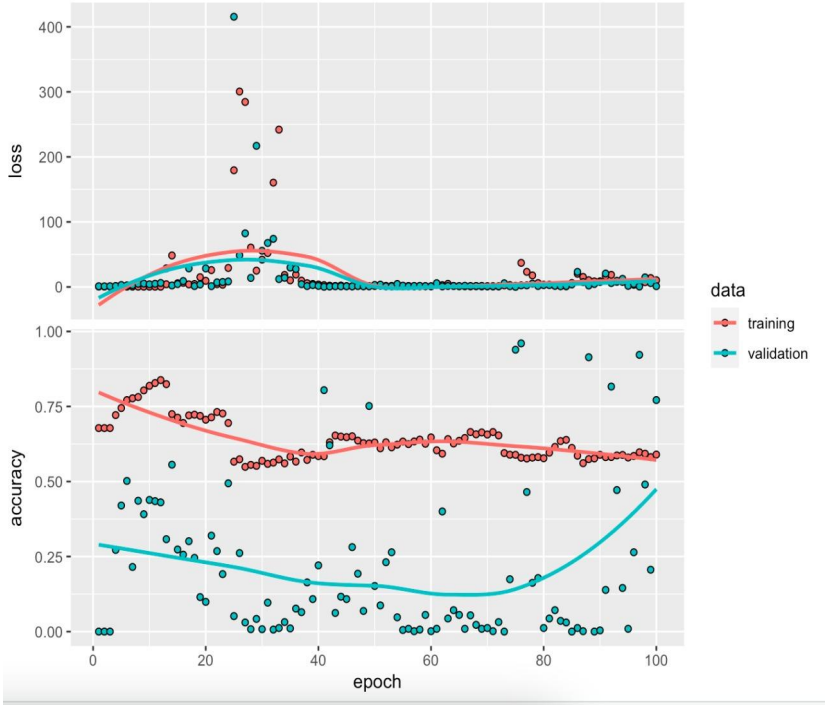
- **Loss: 10.29 y Accuracy 0.58**

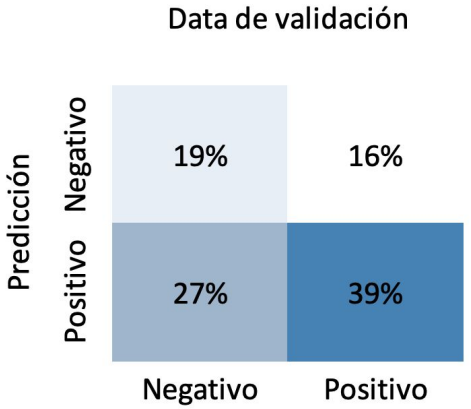
Validation set

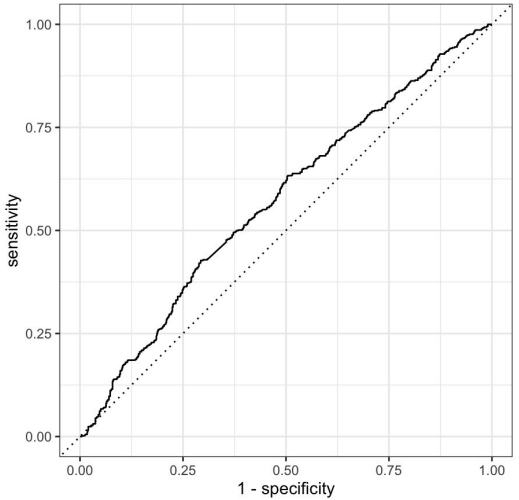
- **Loss: 1.202 y Accuracy 0.7715**

Capas	**Activación**	**Regularizador**	**Unidades**	**Parámetros**
1 Embeddings	Relu	–	32	22432
2 Bidirectional	Relu	Dropout del 40%	64	49664
3 Bidirectional	Relu	Dropout del 40% + L2	32	41216
4 Bidirectional	Relu	Dropout del 40%	16	10368
5 Bidirectional	Relu	Dropout del 40%	32	16640
6 Bidirectional	Relu	–	64	66048
Dense	Sigmoide	–	1	129

LSTM


loss
accuracy
epoch
data
training
validation


Data de validación
Predicción
Negativo
Positivo
19%
16%
27%
39%
Negativo
Positivo


sensitivity
1 - specificity

LSTM

Más características:

- Batch de 20, 50 épocas
- Adam
- binary_crossentropy

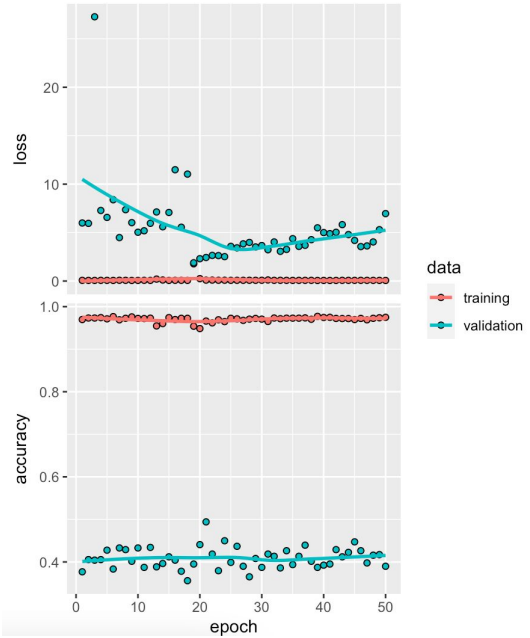
Problema

- El modelo depende de los datos

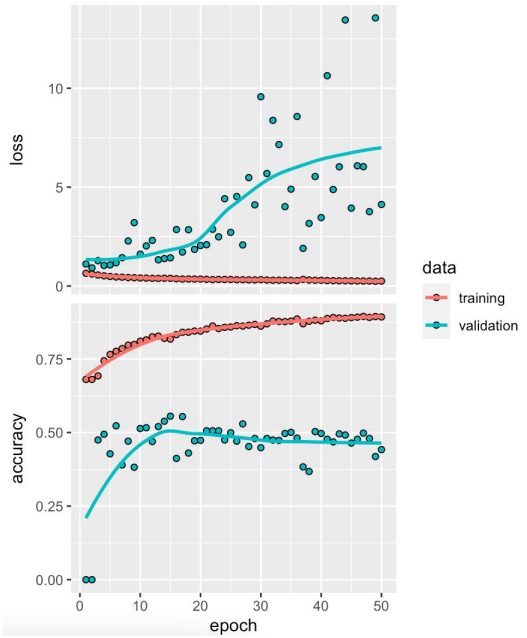
Otras herramientas que intentamos

- Menos/más layers
- Menos/más dropout, L1, L1 y L2
- Otras activaciones
- Capas LSTM unidireccionales
- Batch normalization
- Adagrad, Adadelta
- Quitamos stopwords, dejamos tuits de 1-3 palabras
- Regresamos hashtags

Seed 1



Seed 2



Resultados

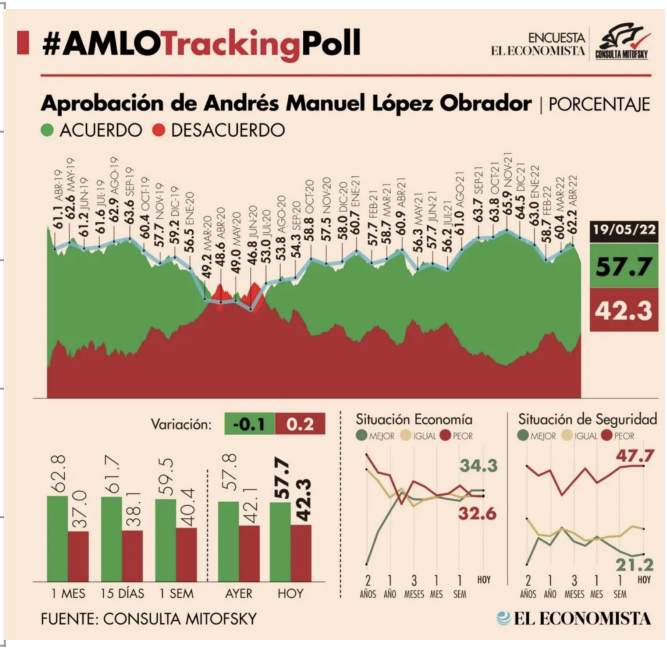
- Realizamos tres tipos de modelos: LSTM, BERT, Multinomial Lasso.
- La mejor precisión en clasificación en el set de validación se obtuvo BERT (aunque en una muestra ganó LSTM).
- Como se anticipaba, los resultados del modelo de ML fueron inferiores a los obtenidos con ambos modelos de DL.
- Un resultado no esperado es que LSTM tuviera mejor desempeño que BETO, aunque el LSTM es muy dependiente de la muestra.

	LSTM	**BETO**	**Lasso**
Validation accuracy	0.7715 (0.46)	0.68	0.52

Indicador semanal

- Se tomaron nuevos Tweets de la semana pasada para generar el Indicador de apoyo a AMLO, los resultados que se obtuvieron fueron los siguientes:

Modelo	Total Negativos	Porcentaje negativos	Total
BETO 4	4565	45.6%	10 mil
BETO 5	4691	46.9%	10 mil
Lasso Multinomial	70,140	74.9%	93,567
LSTM	38681	41.34%	93,567



LSTM vs BERT

- Modelos más básicos como **Word2Vec** o **GloVe** no consideran el contexto de las palabras y tienen dificultades para capturar todos los posibles significados de una palabra. Tienen dificultades adicionales en caso de que la ortografía o la gramática no sean correctas (como en el caso de tweets).

- A diferencia de estos modelos, que usan el token anterior para predecir el siguiente, **BERT** utiliza tanto el token anterior como el siguiente para hacer predicciones.

- La característica clave de **BERT** es que enmascara aleatoriamente las palabras en un contexto dado y realiza predicciones sobre ellas.

- Un **LSTM** bidireccional analiza las palabras en una oración y su contexto de izquierda a derecha y de derecha a izquierda por separado. Al concatenar el embedding de esas palabras, se puede perder el sentido del contexto/posición.

- Esto contrasta con **BERT** que aprende de todas las palabras en la oración de forma simultánea.

- En la tarea de análisis de sentimiento de tweets, podría ser que **BERT** tenga dificultades dado el vocabulario con el que fue entrenado.

- Un **LSTM** podría obtener buenos resultados debido a que la estructura del texto de los tweets es sencilla, bastando un análisis bidireccional. El que la tarea sea de clasificación simple y la longitud de los tweets esté restringida también contribuyen al desempeño de un **LSTM**.

- De acuerdo con Ezen-Can (2020), modelos **LSTM bidireccionales** pueden lograr resultados significativamente mejores a los de **BERT** para una base de datos pequeña. Concluye que el desempeño de un modelo depende de los datos y de la tarea que se quiera realizar.

BETO+LSTM

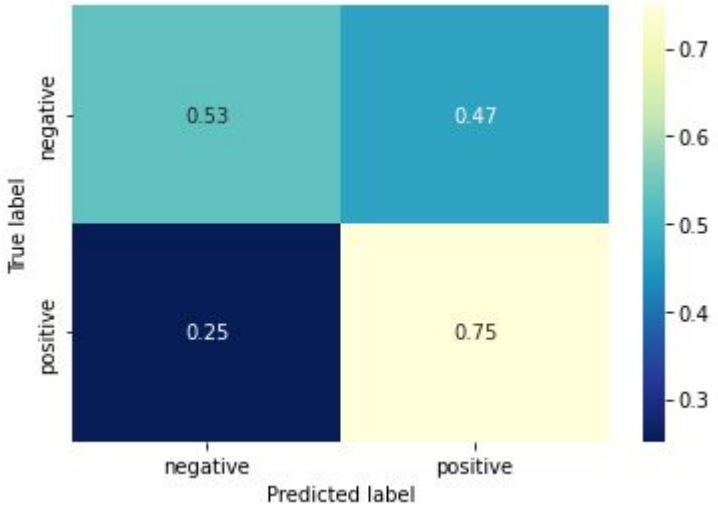
- Hay evidencia de que alimentar un LSTM con los embeddings de un BERT resulta bueno para resolver preguntas (SQUAD challenge) [Zhang, Y. & Xu,Z.]

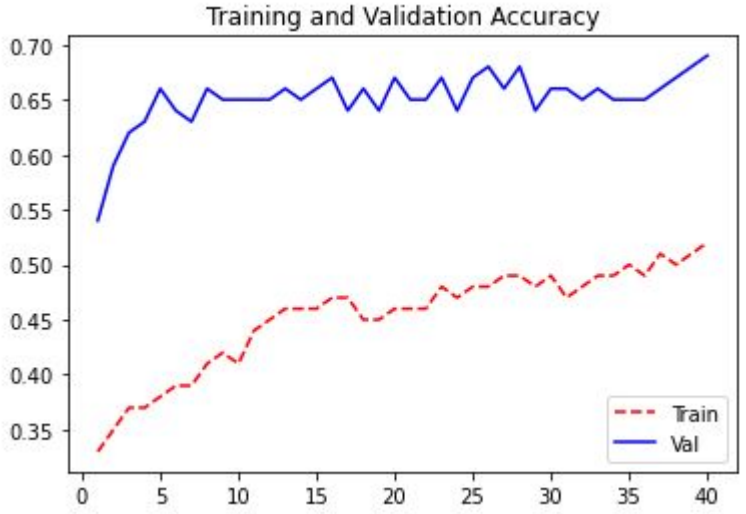
- Corrimos un BERT sin la capa de clasificación y los embeddings los enviamos a un LSTM bidireccional (768x256). Finalmente agregamos una capa Dense con activación Sigmoide (256x2)

- Se hicieron pruebas en un CPU de Google Cloud (4 vCPUs, 16GB RAM) y finalmente se corrieron en un GPU de colab

Modelo	**Clases**	**Optimizador**	**Regularización**	**Pre-Procesamiento**	**Epochs/Time**	**Val Acc.**
BETO+LSTM +Dense	2	Adam (lr=4e-5,eps= 1e-6)	weigth_decay=0.01	max_tokens = 40	30 s p/e (colab GPU) 40 epochs	67%
BETO+LSTM+ Lineal	2	AdamW (lr=4e-4,eps= 1e-5)	weigth_decay=0.01	max_tokens = 40	20 s p/e (colab GPU) 25 epochs	55%

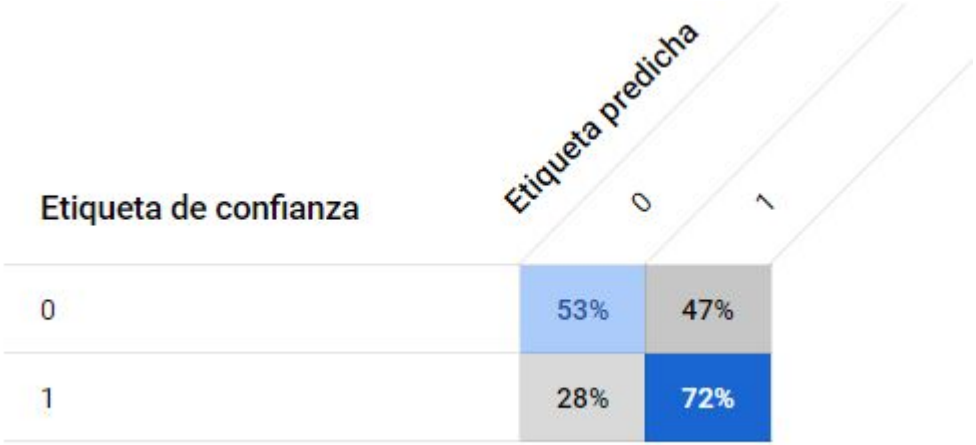
BETO+LSTM (mejor intento)





Modelo AutoML de Google

- Precisión promedio: 0.676
- Tiempo de entrenamiento: 4 h 49 min
- División de datos: Asignado de forma
- aleatoria (80/10/10)
- Algoritmo AutoML



Discusión

- La clasificación se volvió complicada (limitaciones) debido a:
  - Sarcasmo en tweets.
  - Groserías no son un buen indicador para establecer una postura.

- Algunas sugerencias podrían ser:
  - Ampliar la muestra.
  - Balancear la muestra.
  - Tener datos demográficos de usuarios (imposible, pero podría mejorar el desempeño).
  - Agregarle una capa de Contextual CNN encima del BETO [Hulburd, E & Gupta, S.]

- Justificación de diseño:
  - Tener más precisión en las clasificaciones era el objetivo, se intentaron tres modelos y basados en su desempeño decidimos que el modelo que se pondría en producción sería el LSTM bidireccional.

Conclusiones:

- Los métodos de Deep Learning constituyen una herramienta versátil para análisis de lenguaje. Para clasificación de sentimiento son superiores en eficiencia y precisión a las alternativas de hacerlo manualmente o usando métodos de ML.

- A pesar de las limitaciones que surgen de utiliza tweets, un LSTM bidireccional es útil para saber cuál ha sido la aprobación del gobierno en la última semana. En particular para esta semana, esa aprobación fue de 63% (tweets que mencionan al Presidente de manera no negativa).

- Un tratamiento más profundo de los datos puede tener el potencial de incrementar la eficiencia y precisión del modelo.

Referencias

- Cañete, J. et al (2020), Spanish Pre-Trained BERT Model and Evaluation Data, PML4DC at ICLR 2020 available at: https://github.com/dccuchile/beto
- Ezen-Can, A. (2020). A Comparison of LSTM and BERT for Small Corpus. *arXiv preprint arXiv:2009.05451*.
- Hvitfeldt, E., & Silge, J. (n.d.). Supervised machine learning for text analysis in R. Retrieved May 15, 2022, from https://smltar.com
- Singh, A. (2019, June 26). *Building state-of-the-art language models with bert*. Medium. Retrieved May 19, 2022, from https://medium.com/saarthi-ai/bert-how-to-build-state-of-the-art-language-models-59dddfa9ac5d#:~:text=Meaning%20there%20are%20two%20LSTMs,positions%2C%20meaning%20the%20entire%20sentence.
- Sood, S. (2020, December 26). *LSTM & Bert Models for Natural Language Processing (NLP)*. OpenGenus IQ: Computing Expertise & Legacy. Retrieved May 19, 2022, from https://iq.opengenus.org/lstm-bert-nlp-model/
- Takahashi, Y. (2020, November 9). *LSTM vs Bert - a step-by-step guide for tweet sentiment analysis*. Medium. Retrieved May 19, 2022, from https://towardsdatascience.com/lstm-vs-bert-a-step-by-step-guide-for-tweet-sentiment-analysis-ced697948c47
- Zhang, Y. & Xu,Z. BERT for Question Answering on SQuAD 2, work in progress available at: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15848021.pdf

Duran, B. (2020, August 4) Análsis de Sentimiento usando BERT en español, disponible en: https://benjad.github.io/2020/08/04/clasificador-sentimiento-BERT/

Hulburd, E & Gupta, S. (2022) Exploring Neural Net Augmentation to BERT for Question Answering on SQUAD 2.0, available at: https://arxiv.org/ftp/arxiv/papers/1908/1908.01767.pdf