Detección de noticias falsas con Machine Learning

Christian Díaz Guerra
Departamento de Ingeniería Eléctrica
Universidad de Chile

I. Introducción

Con la masificación del uso de internet, el acceso a la información se ha facilitado de gran manera. Hoy en día, basta una búsqueda simple en el celular o computador para estar enterado de lo que está ocurriendo en cualquier parte del mundo. Las redes sociales han tenido una gran influencia en la propagación rápida de las noticias, al permitir informar los acontecimientos de manera prácticamente instantánea, compartirlos con el resto de la comunidad y reaccionar a estos. De hecho, en Chile, la red social Facebook, ha sido de los medios más utilizados por la población para informarse, estando por encima de los más tradicionales, como lo son la radio y periódicos [1].

Sin embargo, esta masificación también ha traído problemas: la desinformación. Esto podría parecer paradójico, pero esta facilidad en la propagación de la información también ha conllevado en la propagación de gran cantidad de noticias falsas. Las redes sociales han tratado de lidiar con esto, detectando este tipo de noticias y eliminándolas. Otras organizaciones también han hecho esfuerzos por desmentir estas informaciones, por medio de lo que se ha denominado *Fact Checking*. Sin embargo, en ocasiones, la noticia se ha compartido tanto que es difícil refutarla [2].

Este problema de la propagación de noticias falsas se ha vuelto bastante serio, e incluso ha sido catalogado como un problema gubernamental en algunos países, debido a la influencia en la política del país. Luego del referéndum de 2016 y el escándalo de *Cambridge Analytica* [3], el parlamento del Reino Unido indagó en estas prácticas y como influyen en el pensamiento político de los ciudadanos [4]. En esta investigación también se indican los diferentes tipos de noticias falsas: los contenidos fabricados, manipulados, suplantados, engañosos, con contexto falso y aquellas satíricas o en forma de parodia.

Debido al problema de la difusión de noticias falsas, este trabajo tiene como objetivo la detección de estas con técnicas de Machine Learning, a saber, Naive-Bayes, Regresión Logística, SVM y Redes Neuronales. Para esto, se han recopilado noticias falsas de sitios web dedicados a la producción satírica de estas, y noticias reales de fuentes confiables de información, creando modelos de clasificación y testeándolos.

En las siguientes secciones se muestra el proceso de procesamiento de los datos, los métodos utilizados, sus resultados y análisis, para terminar con una conclusión y propuesta de trabajo futuro.

II. Marco Teórico

A. Preprocesamiento de los datos

Para poder representar un documento y poder utilizarlo en los modelos de machine learning, este tiene que ser representado de una forma sencilla. Para ello, cada uno de ellos se *tokeniza*, i.e., se separa en trozos llamados *tokens*. Estos *tokens* pueden estar formados por varias palabras (n-gramas), pero en un trabajo similar [5], se han obtenido mejores resultados utilizando *tokens* de un solo término (unigrama). En este proceso, también se convierten todo a minúsculas, pues de esta forma, todas las palabras tienen el mismo formato.

1) Eliminación de stop words: En el español, hay ciertas palabras que son utilizadas en gran frecuencia. Ejemplos de estas son *de, la, que, el, en, y, a, por, un, con, su, etc.* Estos términos no aportan demasiado pues solo sirven para conectar ideas y para estructurar las oraciones, por lo que al eliminarlas, los modelos pueden enfocarse en palabras más importantes y distintivas de cada tipo de documento.

2) Eliminación de símbolos: Los signos de puntuación no aportan demasiado al análisis del documento, por lo que pueden ser eliminados. Además, ciertas noticias presentan *tweets*, por lo que el uso de *# (hashtag)* y @ puede repetirse en varios de ellos. Estos son mejor eliminarlos, sobre todo el uso del formato de Twitter @*usuario*, pues puede aparecer el nombre del sitio web o del periodista asociado a la noticia. También se eliminan los *links* debido a que son demasiado específicos y no aportan demasiado.

3) Stemming: Este proceso consiste en reducir una palabra a su raíz. Para ello, se utilizan algoritmos heurísticos para cortar el final de las palabras. Estos a veces no proveen la verdadera raíz, pero son suficientes para lo que se busca. Así, por ejemplo, las palabras *aprender, aprendiendo, aprendí* se convierten en *aprend*. Esto proceso permite reducir el número de palabras en el vocabulario (que se utiliza al momento de extraer las características).

B. Extracción de características

Para poder utilizar los modelos, los datos tienen que ser convertidos a matrices de números. Para ello, se pueden utilizar las siguientes técnicas [6]

1) Term Frecuency (TF): Dado un documento ya preprocesado, para cada palabra de un vocabulario dado, se cuenta la cantidad de veces que la palabra aparece en dicho documento. Así, este queda representado con un vector de largo igual al del vocabulario, en donde cada componente refiere a este cálculo para cada palabra.

2) TF-IDF: Similar a lo anterior, ahora la frecuencia de cada término se multiplica por su factor *idf*, el cual se define como

$$idf_{t_i} = \log_{10}\left(\frac{N}{n_i}\right)$$

Donde t_i es el término, N es la cantidad de documentos de la colección y n_i es la cantidad de documentos que contienen el término t_i. Con esto, las palabras que aparecen en todos los documentos tienen un factor 0, mientras que las que aparecen en pocos, tienen un mayor valor. Con esto, se le da un mayor peso a palabras específicas de cierto tipo de documentos.

3) Word Embeddings: Esta forma de extraer características permite transformar palabras a un vector de números de un largo dado. Estos son entrenados con gran cantidad de textos para que puedan captar las relaciones entre gran cantidad de palabras. Para ello, existen diversos algoritmos, como *Word2Vec*, *FastText* y *GloVe*.

C. Modelos de Machine Learning

1) Naive-Bayes Multinomial: Este modelo asume que la posición de cada término no importa, además de que cada característica es independiente de las otras [7]. La clase que predice este clasificador viene dado por

$$c_{NB} = arg\max_{c_j \in C} P(c_j)\Pi_{x \in X} P(x|c_j)$$

donde $P(c_j)$ es la probabilidad de que el documento sea de la clase c_j y $P(x|c_j)$ de que el término x pertenezca a esta misma clase. Para calcular estas probabilidades, se usa TF del término en esta clase dividido por la cantidad de documentos de esa clase, y para $P(c_j)$ se usa

$$P(c_j) = \frac{\text{\# documentos de clase } c_j}{\text{\# documentos}}$$

Puede ocurrir que un término no aparezca en el documento, lo que haría su probabilidad igual a 0 y que todo el documento tenga esta misma probabilidad de ser de la clase c_j, aún cuando es de esta clase. Para evitar esto, al calcular TF, se puede realizar lo que se llama *smoothing*, el cual viene dado por

$$P(x|c_j) = \frac{(\text{\# de ocurrencias de x en clase j}) + \alpha}{(\text{\# documentos clase j}) + \alpha|V|}$$

donde $|V|$ es la cantidad de elementos del vocabulario.

2) Regresión Logística: En el caso de clasificación binaria, para calcular la probabilidad de que un documento x sea de clase C_1, la regresión logística [8] se basa en la ecuación

$$P(C_1|x) = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)}$$

en donde se debe encontrar el w. Para ello se puede optimizar la log-verosimilitud, cuyo valor está dado por

$$l(w) = \sum_{i=1}^{N} y_i \log(\sigma_i) + (1 - y_i)\log(1 - \sigma_i)$$

donde y_i es la clase del dato i, y $\sigma_i = \sigma(w^T x_i)$. Para predecir, se calcula $\sigma(w^T x)$ y si es > 0.5, es clase 1, y clase 0 si no.

3) SVM: Una Máquina de Soporte Vectorial (SVM por sus siglas en inglés) se basa en la idea de encontrar el mejor hiperplano que divida un conjunto de datos en dos clases. Este modelo tiene parámetros que pueden ser ajustados para un mejor ajuste. Estos son el parámetro C, el cual al tener un valor más alto, provoca que el modelo sea más insesgado, pero con una mayor varianza y lo contrario al ser pequeño; y el tipo de *kernel*, el cual es una función matemática que permite separar datos que no son linealmente separables (si se utiliza un *kernel* no lineal). Para el caso del kernel RBF, se necesita un parámetro adicional denominado γ (gamma), el cual determina el nivel de curvatura de la región de decisión. Mientras más alto su valor, mayor curvatura.

4) Red Neuronal Recurrente: Una red neuronal recurrente (RNN) es un tipo de red neuronal que permite conexiones entre neuronas, creando temporalidad y permitiendo a la red, tener memoria. Son ampliamente utilizadas en el análisis de secuencias, como texto, audio o video. Hay varios tipos, siendo una de ellas las LSTM (long short term memory). En algunos tipos, la acumulación de errores dificulta la memorización a largo plazo. La LSTM solventa este problema.

D. Métricas de evaluación

1) Accuracy: Esta métrica mide la cantidad de predicciones correctas que tuvo el modelo. Para ello, calcula

$$\text{Accuracy} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$$

El modelo tiene mejor desempeño mientras el valor sea más cercano a 1.

2) Matthews correlation coefficient (MCC): Para esta métrica se utiliza la siguiente ecuación

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Donde $TP =$ cantidad de verdaderos positivos, $TN =$ cantidad de veraderos negativos, $FP =$ cantidad de falsos positivos y $FN =$ cantidad de falsos negativos. Este puede tener como peor valor -1, lo que indica una completa clasificación errónea (está clasificando negativos como positivos y viceversa) y como mejor valor 1, lo cual indica perfecta clasificación. Un valor igual a 0 es el valor esperado de lanzar una moneda. Esta métrica resuelve algunos problemas que se pueden presentar con conjuntos desbalanceados y es preferible por sobre otras formas de medición como F1-score [9].

III. Dataset

Para este trabajo, no se logró encontrar un repositorio público con noticias reales y falsas en español, por lo que se recolectaron desde distintos sitios web abiertos. Así, para las noticias falsas, se utilizaron las páginas La Legal (Chile), El Mundo Today (España), Hay Noticia (España), El Chigüire Bipolar (Venezuela) y Última Noticia (Perú). Para el caso de las noticias reales, se rescataron de los portales El Mostrador (Chile), El Mundo (España) y El Universo (Ecuador). Hay que notar que en el primer portal también se publican noticias de otros medios, como DW y EFE, ambas de carácter internacional. En la Tabla I se muestra la cantidad de cada una de ellas.

TABLE I
CANTIDAD DE NOTICIAS RECOLECTADAS DE CADA SITIO WEB

	Sitio	Cantidad	Total
Falsas	El Chigüire Bipolar	1344	22938
	El Mundo Today	6701	
	Hay Noticia	2491	
	La Legal	12121	
	Última Noticia	281	
Reales	El Mostrador	13652	17768
	El Mundo	2758	
	El Universo	1358	

IV. METODOLOGÍA

El dataset se particiona en conjuntos de entrenamiento, validación y testeo. Para el primer y segundo conjunto, se usan las noticias de los sitios El Mundo Today, Hay Noticia, La Legal, El Mostrador y El Mundo, con una proporción de 85% y 15% respectivamente. Para el caso del conjunto de testeo, se utilizan las demás noticias. Esto se realiza para que las pruebas sean más reales, pues así se podrá ver si los modelos sirven para otro tipo de editorial y otros países. Hay que notar que las clases están desbalanceadas, por lo que la mejor forma de evaluación es usando MCC.

A estos conjuntos se le aplican los modelos de la sección II-C, sobre los títulos y cuerpos de cada noticia de forma separada. Para el caso de los tres primeros modelos, se realizan todas las técnicas de preprocesamiento dadas en la sección II-A. Estos permiten que todas las noticias se encuentren en un formato estándar y que el vocabulario sea más acotado, para que el entrenamiento no tenga tanto costo computacional. Sin embargo, para la *red neuronal*, no se hace *stemming*, debido a que se utilizarán word embeddings pre entrenados para representar cada palabra, por lo que se necesitan que estas se encuentren en la misma forma como aparecen.

En Naive-Bayes, por su naturaleza, se utiliza TF como método de extracción de características. Para regresión logística y SVM, se utiliza TF-IDF, pues con esto se tiene una mejor forma de representar el peso de cada palabra en el documento. Hay que tener en cuenta que el vocabulario y el factor IDF de cada palabra se obtienen solo del conjunto de entrenamiento, pues es de este del que se conoce todo, los demás, en teoría, son desconocidos. Para el caso de la red neuronal, se utilizan word embeddings producidos con el algoritmo *FastText* y proporcionado por José Cañete [10], con una dimensión de 10. Se podría elegir una dimensión mayor, pero se encontró que no era necesario y que al elevarlo, el costo computacional podría ser muy alto, sobre todo para el conjunto de cuerpos de noticias.

Para el caso de Naive-Bayes, se utiliza el parámetro $\alpha = 1$, conocido como *Laplace Smoothing*, pues con esto, una palabra no puede tener una probabilidad igual a 0, y la clase tampoco. Para SVM se utilizan valores de $C \in \{0.1, 1, 10\}$, y kernels lineal y RBF, esto para encontrar la combinación que dé los mejores resultados. Para este último caso, además se utilizan los valores de gamma $\gamma \in \{0.1, 1\}$.

Para el caso de la RNN, esta se implementa con las capas Embedding, la cual no tiene parámetros entrenables, debido a que se utilizan word embeddings ya entrenados; Bidirectional con LSTM con 64 neuronas, pues las LSTM's permiten capturar secuencias, en este caso, secuencia de texto, y la bidireccional hace que se analice hacia ambos sentidos, permitiendo una mejor comprensión de la noticia; Dropout(0.5), para que el modelo no se sobre entrene, eliminando aleatoriamente conexiones entre neuronas; y finalmente una Dense, con una sola neurona, con activación sigmoide, para que entregue un resultado entre 0 y 1, que son los valores en los que están codificados cada clase.

Tanto para la extracción de características (TF y TF-IDF) como para la implementación de los tres primeros modelos y métricas, se utiliza el paquete de *sklearn* que permite llevarlos a cabo. Para el caso de la red neuronal, se utiliza la biblioteca *keras*.

TABLE II
RESULTADOS DE LOS MODELOS EN EL CONJUNTO DE VALIDACIÓN

	Título		Cuerpo	
Modelo	**Accuracy**	**MCC**	**Accuracy**	**MCC**
Naive-Bayes	0.88	0.76	0.94	0.88
Regresión Logística	0.88	0.77	0.97	0.95
SVM lineal, C=1	0.89	0.78	0.98	0.95
SVM RBF, C=10, γ = 1	0.90	0.80	0.98	0.96
RNN	0.85	0.70	0.85	0.69

TABLE III
RESULTADOS DE LOS MODELOS EN EL CONJUNTO DE TESTEO

	Título		Cuerpo	
Modelo	**Accuracy**	**MCC**	**Accuracy**	**MCC**
Naive-Bayes	0.66	0.33	0.79	0.59
Regresión Logística	0.67	0.34	0.89	0.78
SVM lineal, C=1	0.68	0.35	0.89	0.79
SVM RBF, C=10, γ = 1	0.68	0.35	0.90	0.80
RNN	0.72	0.45	0.62	0.24

V. RESULTADOS Y ANÁLISIS

En las Tablas II y III se muestran los resultados de los distintos modelos sobre el conjunto de validación y testeo respectivamente. Se reporta su *accuracy* y *MCC*. Para el caso de SVM, solo se muestran los modelos con los parámetros que dieron los mejores resultados en el conjunto de validación.

Se observa que para el conjunto de validación, el mejor resultado lo obtuvo el modelo SVM con un kernel RBF con los parámetros $C = 10$ y $\gamma = 1$, tanto al analizar los títulos como los cuerpos de las noticias. Para este último caso se observa que el accuracy es de un 98% y $MCC = 0.96$. Para el caso del conjunto de testeo, este modelo también da el mejor resultado al analizar el cuerpo, con un *accuracy* de 90% y $MCC = 0.8$. Sin embargo, para el caso de los títulos, el mejor resultado lo da la RNN, aún cuando para los cuerpos su desempeño es pobre, con un *accuracy* de 72% y $MCC = 0.45$.

En ambos casos, tanto en el conjunto de validación como en el de testeo, se observa que los modelos funcionaron correctamente, logrando un buen accuracy y un alto MCC. Para el caso de los títulos, no hay una gran diferencia en los resultados, excepto para el caso de la RNN. En el caso del análisis del cuerpo de la noticia, Naive-Bayes y RNN se comportan mucho peor en comparación con los demás modelos.

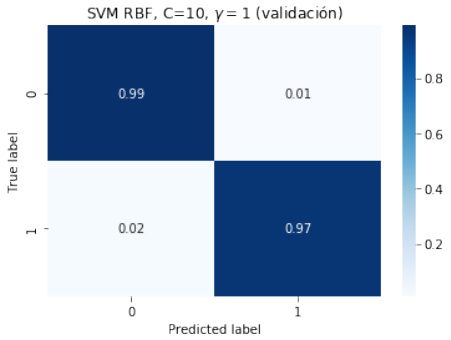

Fig. 1. Matriz de confusión conjunto de validación modelo SVM RBF en cuerpo

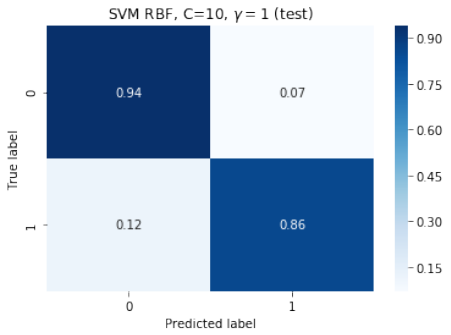

Fig. 2. Matriz de confusión conjunto de testeo modelo SVM RBF en cuerpo

En las figuras 1 y 2 se muestran la matriz de confusión normalizada (redondeada a 2 decimales) del mejor modelo (SVM) sobre los cuerpos, para el conjunto de validación y testeo respectivamente. Se observa que en ambos casos, este tiene mayor porcentaje de aciertos en las noticias falsas, lo cual coincide con que en el conjunto de entrenamiento habían más cantidad de este tipo.

Aún cuando el modelo SVM fue el que obtuvo mejor resultado, el costo computacional de este es bastante elevado. Regresión logística tiene resultados similares con un tiempo de procesamiento mucho menor, lo que podría ayudar al tener un conjunto de noticias muy grande.

Además, se observa que aunque al aplicar los modelos sobre los títulos en el conjunto de validación, los resultados fueron buenos, en el conjunto de testeo no lo son, por lo que solo al considerar el encabezado, estos no pueden adaptarse a noticias de otros sitios web o países tan bien como lo hace el modelo sobre el cuerpo de estas.

VI. Conclusión

A partir de los resultados obtenidos, se puede afirmar que el objetivo de clasificar noticias en reales y falsas, es posible de llevarse a cabo con técnicas de Machine Learning. Para esto, la mejor forma es utilizando un modelo SVM con kernel RBF y parámetros $C = 10, \gamma = 1$, con preprocesamiento de las noticias eliminando stopwords, dejando todo en minúsculas y haciendo stemming a las palabras, con extracción de características dados por TF-IDF.

Sin embargo, una de las dificultades de este modelo es su alto costo computacional. Tanto el entrenamiento como la predicción demora bastante en procesar. Una alternativa es simplemente utilizar regresión logística, la cual obtuvo resultados bastante similares y con un costo mucho menor.

Aún cuando los modelos se comportaron de buena forma tanto al analizar títulos como el cuerpo de la noticia, es mejor utilizar esto último, pues asegura un mejor desempeño incluso para noticias de otras editoriales y países.

Como trabajo futuro, se propone entrenar la RNN con otros parámetros y capas, que den mejores resultados, pues el desempeño obtenido no fue bueno y podrían ser mejores debido a que es un modelo más sofisticado. Aún así, puede que no esté a la altura de SVM, que aún cuando se podría pensar que es más sencillo, puede que tenga todo lo necesario para clasificar correctamente.

Además, sería bueno utilizar más técnicas de Machine Learning para la clasificación, como Decision Trees, k-Nearest Neighbour u otras redes neuronales, y ver si se obtienen mejores resultados.

Otra modificación que podría realizarse es utilizar otro set de datos. Las noticias falsas utilizadas aquí son satíricas y no son las que más daño hacen al compartirse, pues una persona puede darse cuenta no tan difícilmente de que es falsa (aún así ha ocurrido polémicas por este tipo de noticias). Sería bueno obtener y utilizar aquellas que presentan contenido fabricado, manipulado o engañoso, y que para un humano sea difícil de decidir si es real o falsa.

References

[1] Cadem, “Estudio Medios de Comunicación post crisis.” Santiago, 16-Jan-2020.
[2] M. Alonso González, “Fake News: desinformación en la era de la sociedad de la información.,” Ambitos, no. 45, pp. 29–52, 2019, doi: 10.12795/ambitos.2019.i45.03.
[3] C. Boerboom, “Cambridge Analytica: The Scandal on Data Privacy,” Augustana Center for the Study of Ethics Essay Contest, 2020.
[4] UK Parliament, “Disinformation and ‘fake news’: Interim Report Contents”, Jul. 29, 2018. [Online]. Available https://publications.parliament.uk/pa/cm201719/cmselect/cmcumeds/363/36304.htm
[5] H. Ahmed, I. Traore, and S. Saad, “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques,” in Lecture Notes in Computer Science, Springer International Publishing, 2017, pp. 127–138.
[6] F. Bravo, “Vector Space Model and Information Retrieval”, presented to CC6205 Natural Language Processing, Universidad de Chile, Mar. 22, 2020. [PowerPoint slides] Available: https://github.com/dccuchile/CC6205/blob/master/slides/NLP-IR.pdf, Accessed on: Aug. 13, 2020
[7] D. Jurafsky, “Text Classification and Naive Bayes”, presented to Speech and Language Processing, Standford, 2019. [PowerPoint slides] Available: https://web.stanford.edu/ jurafsky/slp3/slides/7_NB.pdf, Accessed on: Aug. 13, 2020
[8] F. Tobar. Aprendizaje de Máquinas [Online]. Available: https://github.com/GAMES-UChile/Curso-Aprendizaje-de-Maquinas/blob/master/notas_de_clase.pdf
[9] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”, BMC Genomics, vol. 21, no. 1, Jan. 2020, doi: 10.1186/s12864-019-6413-7.
[10] José Cañete, “Spanish Word Embeddings”. Zenodo, 30-May-2019