

**Universidad De Chile**

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

# LDA CON DATOS DE TWITTER

Javier Castro Medina

Abril 2021

# 1 Análisis Exploratorio

Para la obtención de tweets se usa la librería `twint`<sup>1</sup>. Esta librería es simple de usar, permite obtener una gran cantidad de datos y almacenarlos en formato `csv` u otros. En las Figuras 1a y 1b se muestra, a grandes rasgos, el código que se necesita para obtener los datos utilizados en este trabajo.

```
# Define c object
c = twint.Config()
# Parameters
c.Search = word
c.Near = 'Santiago, Chile'
c.Since = since
c.Until = until
c.Store_csv = True
c.Output = 'data/' + file_name + '.csv'
# Run search
twint.run.Search(c)
```

(a)

```
# Define c object
c = twint.Config()
# Parameters
c.Username = id
c.Store_csv = True
c.Output = 'data/' + id + '.csv'
# Run search
twint.run.Search(c)
```

(b)

Figure 1: A la izquierda se buscan y guardan tweets creados al rededor de Santiago entre las fechas `since` y `until` que además contengan la palabra `word`. A la derecha se obtienen y guardan los tweets de la cuenta `id`. **Scrapping tweets** es la expresión en inglés para este proceso.

En la primera parte del proyecto se realiza un proceso de adaptación a la librería `twint` y el formato de los datos que permite obtener además de un análisis superficial a los datos que obtenemos en esta fase inicial. Se decide trabajar sólo con tweets en Santiago, esto porque en tal sector hay una población más grande y por lo tanto, una probabilidad más alta de encontrar cuentas de Twitter con respecto a otros lugares. Lo que queda de esta sección se divide en dos partes, en la primera se mostrarán wordclouds y en la segunda gráficos de la frecuencia de tweets diarios para distintos set de datos.

**Remark 1.1.** *Otra razón por la cual fijar el lugar geográfico de la obtención de tweets es disminuir la complejidad del proyecto, puede ser un aspecto a considerar en una tarea de mayor escala en la que además se tome en cuenta la variable espacial de nuestros datos.*

## 1.1 Wordclouds

Se muestran las wordclouds asociadas a tres set de tweets (o datos) distintos. Por un lado tenemos dos hitos importantes en Chile, el primero es el estallido social del 18 de octubre del 2019 y el segundo es el día de la mujer del 2020. El tercero contiene los tweets que contienen la palabra “piñera” y fueron creados durante el 2020.

---

<sup>1</sup>[github.com/twintproject/twint](https://github.com/twintproject/twint)

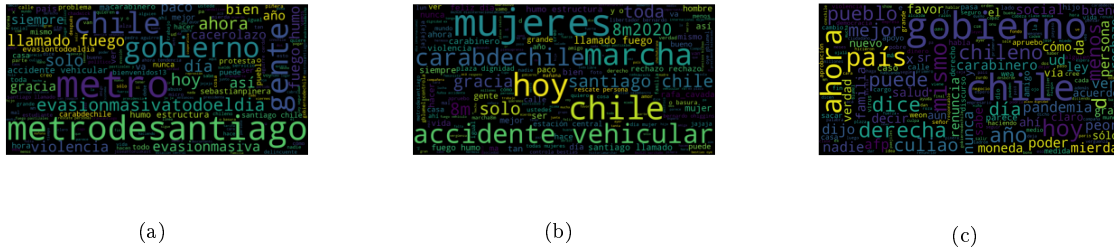


Figure 2: De izquierda a derecha: estallido social, día de la mujer y tweets que contienen la palabra “piñera”.

Para encontrar los tweets del estallido y el día de la mujer sólo se especificaron las fechas y el lugar, el campo `c.Search` que aparece en la Figura 1a se deja vacío. En cambio, para los tweets con “piñera”, se debe usar `c.Search = 'piñera'`. Las palabras **metrodesantiago** y **evasionmasivatodoeldía** que aparecen en 2a corresponden a hashtags que estuvieron presentes en Twitter durante los días previos al estallido. En la Figura 2b observamos las palabras **mujeres** y **marcha**, esto sucede porque en esa fecha ocurre la marcha en conmemoración del día de la mujer que, como vemos, tiene impacto en redes sociales como Twitter. En la Figura 2c se destacan las palabras **gobierno**, **Chile** y **derecha** que se relacionan bastante con lo que representa la palabra “piñera”. En las tres wordclouds se encuentran palabras bastante esperables y que guardan estrecha relación con el contexto social en el cual fueron creados los tweets, esta es una señal que confirma que nuestros datos tienen sentido con su variable temporal.

## 1.2 Series de tiempo

Se exponen series de tiempo relativas a la cantidad de tweets diarios que contengan cierta palabra clave. Acá trabajamos con dos set de tweets, por un lado tenemos uno de la sección anterior, asociado a la palabra “piñera”, y por otro contamos con los tweets que contienen la palabra “covid” y que fueron creados durante el año 2020.

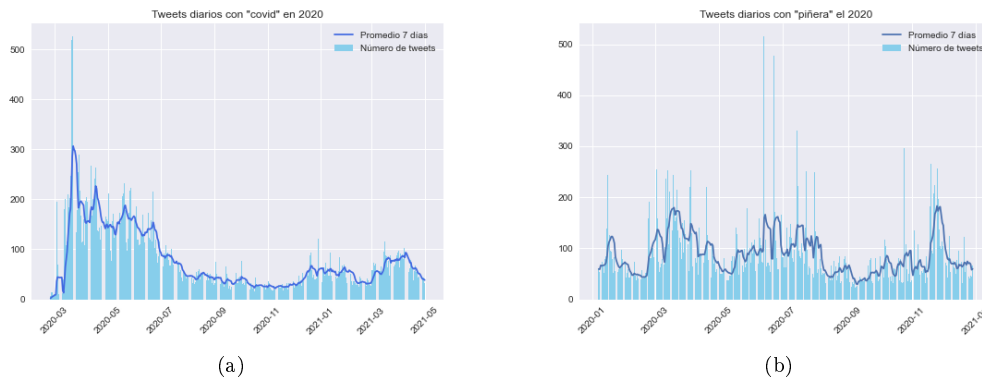


Figure 3: Cantidad de tweets diarios con la palabra “covid” (izq) y “piñera” (der).

En la Figura 3a vemos que la curva que se genera al suavizar tomando media móvil de siete días es similar a la curva de casos diarios de covid en Chile (también suavizada), pero adelantada en tiempo. El primer peak de los casos diarios ocurrió al rededor de junio <sup>2</sup>, mientras que en 3a esto ocurre en marzo cuando todo estaba comenzando. Aproximadamente, en ambas curvas, durante los siguientes seis meses se ve una tendencia a la baja. Esta especie de caracter predictivo de la curva 3a no es sostenible en el tiempo porque si bien los casos aumentaron hasta superar el primer peak, en Twitter durante enero sólo se ve un aumento relativamente pequeño en la cantidad de tweets asociados al covid. Por otra parte, el peak más alto en 3a corresponde al día 23 de marzo del 2020, en esta fecha se informa la primera muerte por covid en Chile.

Con respecto a la Figura 3b, observamos que la cantidad de tweets que hablan de Piñera es relativamente constante en el tiempo, con un promedio de 86 tweets diarios, y sin una tendencia marcada.

---

<sup>2</sup>Link a los casos diarios covid en Chile según Google.

## 2 Latent Dirichlet Allocation aplicado a datos de Twitter

Este algoritmo cae en la categoría del **Topic Modelling** en la cual se busca, a partir de grandes data sets de texto, encontrar tópicos latentes en estos cosa que sería imposible de hacer humanamente. Para este modelo se necesita un set de palabras  $V$  y de tópicos  $K$ , que sin pérdida de generalidad denotamos como  $K = \{1, \dots, K\}$ . Los tópicos se modelan como distribuciones de probabilidad sobre el conjunto de palabras  $V$ . Si bien este algoritmo tiene una fase generativa, su objetivo es calibrar parámetros  $\alpha \in [0, 1]^K$  y  $\beta \in [0, 1]^{K \times V}$  optimizando cierta cantidad. La idea es, dados  $\alpha$  y  $\beta$ , generar palabras con el siguiente algoritmo:

- Se samplea una distribución sobre tópicos  $\theta \sim \text{Dir}(\alpha, K)$ . Notar que  $\sum \theta_{i=1}^K = 1$ . *Dir* es la distribución Dirichlet.
- Dado  $\theta$ , se samplea un tópico  $t \sim \theta$ .
- Dado el tópico  $t$ , se samplea una palabra  $w \sim \beta_t$ .

**Remark 2.1.** *Este algoritmo nos permite interpretar los parámetros del modelo.  $\theta \in \mathbb{R}^K$ , que sólo depende de  $\alpha$ , corresponde a que tan relevante es cada tópico.  $\beta \in [0, 1]^{K \times V}$  modela como se distribuye cada tópico sobre el set de palabras.*

Lo anterior permite samplear una (1) palabra y tomamos los sampleos de forma independiente. Consideremos ahora una palabra  $w_n$ , luego, la probabilidad de generar esta palabra se puede calcular como,

$$\begin{aligned} p(w_n | \alpha, \beta) &= \sum_{t \in K} p(w_n | \alpha, \beta, t) p(t | \alpha, \beta) = \sum_{t \in K} p(w_n | \beta, t) \int p(t | \alpha, \beta, \theta) p(\theta | \alpha, \beta) d\theta \\ &= \sum_{t \in K} p(w_n | \beta, t) \int p(t | \theta) p(\theta | \alpha) d\theta. \end{aligned}$$

Donde usamos probabilidades totales y que, según nuestro modelo,  $p(w_n | \alpha, \beta, t) = p(w_n | \beta, t)$ ,  $p(t | \alpha, \beta, \theta) = p(t | \theta)$  y  $p(\theta | \alpha, \beta) = p(\theta | \alpha)$ . Usando la independencia vemos que para  $w = (w_1, \dots, w_N)$ ,

$$p(w | \alpha, \beta) = \int p(\theta | \alpha) \left( \prod_{n=1}^N \sum_{t \in K} p(w_n | t, \beta) p(t | \theta) \right) d\theta.$$

El calculo anterior tiene razones Bayesianas de ser, esto porque permite calcular la distribución a posteriori de  $(\theta, t)$  la cual se busca maximizar. La distribución que se obtiene es intratable numéricamente por lo cual se sigue un approach MCMC o **variational Bayes** ([1], [2]). La implementación de este algoritmo en **Python** esta dada por la librería **gensim**<sup>3</sup>, el código se basa en el script **onlinedavb.py**<sup>4</sup> de los autores de [2]. En el citado trabajo, los hiperparámetros cambian un poco asumiendo que la distribución  $\beta_t$  de cada tópico  $t \in K$  sobre las palabras  $V$ , es tal que  $\beta \sim \text{Dir}(\eta, |V|)$ . Así, los hiperparámetros (o priors) pasan a ser  $(\alpha, \eta)$  y de esta forma son tratados por **gensim**.

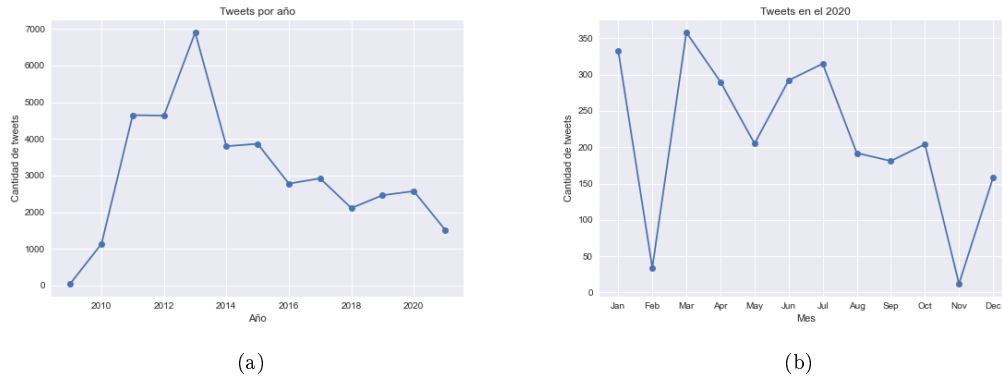
### 2.1 Datos

Podríamos aplicar el modelo a todos los tweets de cierto espacio y tiempo, pero se decide por otro approach que es trabajar con los tweets de una cuenta en específico dado que el rango de temas o tópicos tocados por una cuenta es menor o más controlado que al trabajar con tweets de muchas

<sup>3</sup>Lda con gensim (link)

<sup>4</sup>Link al repositorio.

cuentas a la vez. La cuenta que se escoge para partir es @gabrielboric, obtenemos sus tweets hasta junio del 2021. Las características de este data set se pueden apreciar en las Figuras 4a, 4b y 4c.



<b>Total de tweets</b>	39405
<b>Mínimo</b>	0
<b>Máximo</b>	131
<b>Promedio</b>	9.14

(c)

Figure 4: Tweets por año (a) y por mes en el 2020 (b). Tamaño del data set y estadísticas de la cantidad diaria de tweets (c).

## 2.2 Limpieza y preprocesamiento de datos

Los datos que se tienen disponible corresponden a tweets en formato de texto. El procesamiento de base o inicial que se realiza sobre este corpus corresponde a eliminar **stop words**, links, palabras de largo menor o igual a 3, caracteres indeseables<sup>5</sup> y finalmente pasar todas las palabras a minúscula. El procesamiento anterior se realizará siempre y al inicio de todo código, luego de este proceso a nivel de vocabulario, se decide eliminar aquellos tweets con una cantidad de palabras menor a 5 lo que nos deja con 24218 tweets. Lo siguiente que se puede hacer es quitar palabras de baja frecuencia, sin quitar estas palabras el diccionario<sup>6</sup> resultante tiene un tamaño de 35892 palabras. En el gráfico a continuación muestra la cantidad de palabras resultantes al variar este parámetro.

<sup>5</sup>Se eliminaron: #, comas, signos de exclamación y pregunta, paréntesis y el signo igual.

<sup>6</sup>Entendemos por diccionario al conjunto de todas las palabras utilizadas en el corpus

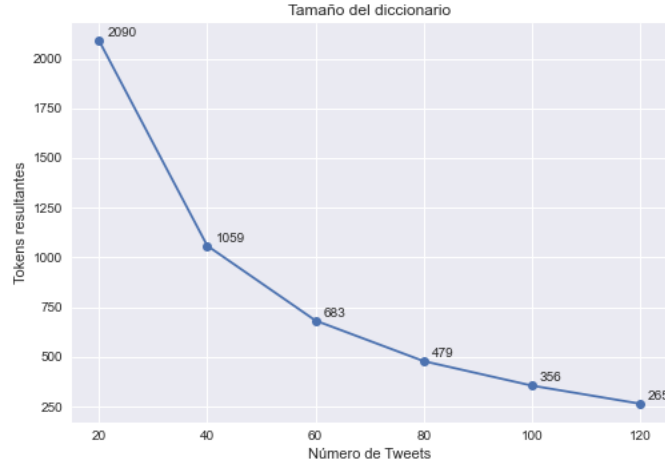


Figure 5: Tamaño del diccionario en función de la cantidad de tweets mínima a la que las palabras deben permanecer. Es decir,  $y$  corresponde a las palabras que resultan al filtrar las que aparezcan en menos de  $x$  tweets. Este gráfico es importante porque nos permite ver con qué tanto vocabulario nos quedamos cuando eliminamos palabras sin mucha importancia o para el modelo.

La siguiente etapa en el preprocesamiento es transformar los tweets en vectores. Para esto, a cada palabra del diccionario, digamos  $V$ , se le asigna un único índice de forma que el conjunto de palabras (o diccionario) se modela como  $\{1, \dots, |V|\}$ . Luego, cada tweet, entendido como una colección de palabras en  $V$  de la forma  $t = \{v_1, \dots, v_n\}$ , se transforma en  $\{(v_1, r_1), \dots, (v_n, r_n)\}$  donde  $v_i$  es el índice de la palabra en  $V$  y  $r_i$  la cantidad de veces que  $v_i$  aparece en  $t$ . A continuación, un ejemplo de esto.

```
bow tweet 304:
[(256, 1), (343, 1), (653, 1), (662, 1), (755, 1), (1078, 1), (1340, 1), (1457, 1), (1646, 1),
(1650, 1), (1805, 1), (1862, 1), (1870, 1), (2263, 1), (2264, 1), (2295, 1), (2434, 1), (2435, 1),
(2438, 1), (2454, 1), (2455, 1), (2456, 1)]
bow tweet 666:
[(4, 1), (10, 1), (273, 1), (445, 1), (492, 1), (733, 1), (752, 1), (769, 2), (1008, 1), (1011, 1),
(1471, 1), (2274, 1), (4151, 1), (4152, 1), (4153, 1), (4154, 1)]
```

Figure 6: Estado final de dos tweets escogidos al azar. Esta representación se conoce como “bow” (**bag of words**).

**Remark 2.2.** *Notar que en los ejemplos que se muestran en la Figura 6, la segunda componente de las tuplas es mayoritariamente 1. Esto pasa porque el largo de cada tweet en el corpus es relativamente pequeño en comparación a otros set de datos como por ejemplo una colección de libros o notas periodísticas. Esto sugiere que esta variable no es muy decisiva en este contexto, o bien, que podríamos juntar tweets que estén relativamente cercanos en tiempo, asumiendo que esto implica parentesco temático, para así generar super-tweets de largo mayor.*

**Remark 2.3.** *Se decide dejar las palabras que hacen referencia a cuentas de Twitter, por ejemplo @javier. La razón de esta decisión es que la aparición de una cuenta en un tópico nos indica que tal persona es relevante para la cuenta sobre la que se está trabajando.*

## 2.3 Aplicando el modelo

Como se comentó en la Sección 2, para aplicar el modelo usamos la implementación de `gensim`. El código de la Figura 7 muestra en términos generales lo necesario para correr el algoritmo sobre nuestro corpus.

```
lda_model = LdaModel(  
    corpus=bow_corpus,  
    id2word=dictionary,  
    alpha='auto',  
    eta='auto',  
    num_topics=5,  
    random_state=1  
)  
  
top_topics = lda_model.top_topics(corpus=bow_corpus, coherence='u_mass')
```

Figure 7: `corpus` corresponde a lo mostrado en la Figura 6, `id2word` es nuestro vocabulario  $V$ , `alpha` y `eta` son los priors discutidos previamente, `num_topics` es la cantidad de tópicos que estamos buscando y `random_state` es la semilla del algoritmo. El método `top_topics` permite obtener coherencia de cada tópico.

El parámetro  $\alpha$  lo podemos entregar como `symmetric`, si queremos que sea un vector, calculado internamente, con todas sus componentes iguales, `asymmetric`, si queremos que sea un vector como el anterior salvo que sus componentes son distintas, `auto` si queremos que el algoritmo calcule un vector tipo `asymmetric` internamente y un escalar  $x$  si queremos que sea tipo `symmetric` pero que el valor que se repita sea  $x$ . Por otro lado,  $\eta$  puede ser todo lo anterior salvo `asymmetric`. Acá es importante mencionar que el campo `coherence` del método `top_topics` admite las métricas denominadas: `u_mass`, `c_v`, `c_uci` y `c_npmi`. De estas, la única que se pudo implementar sin problemas fue `u_mass`, el resto arrojaba un error de Python tipo `Broken Pipe` que no pudo ser solucionado en este proyecto.

### 2.3.1 Primera iteración

En la Figura 8 se muestra el output de pedirle los tópicos al modelo. En esta primera iteración no estamos filtrando palabras de baja frecuencia y los hiperparámetros  $(\alpha, \eta)$  se dejan en default y pedimos 5 tópicos.

```
TOPIC: 0  
WORDS: 0.008*"política" + 0.006*"chile" + 0.006*"tolerancia0" + 0.006*"movimiento" + 0.005*"mejor" +  
0.005*"izquierda" + 0.005*"educación" + 0.005*"derecha" + 0.005*"columna" + 0.004*"recomiendo"  
TOPIC: 1  
WORDS: 0.006*"@jparedesgodoy" + 0.005*"abrazo" + 0.005*"puede" + 0.004*"saludos" +  
0.004*"@valenzuelalevi" + 0.004*"marcha" + 0.004*"alguien" + 0.004*"política" + 0.004*"quiere" +  
0.004*"mismo"  
TOPIC: 2  
WORDS: 0.010*"fech" + 0.009*"aguante" + 0.008*"chile" + 0.008*"universitario" + 0.007*"magallanes" +  
0.007*"buena" + 0.007*"mañana" + 0.007*"estudiantil" + 0.006*"cabros" + 0.006*"@izqautonoma"  
TOPIC: 3  
WORDS: 0.009*"muchas" + 0.008*"ahora" + 0.008*"@matiasdelrio" + 0.008*"vamos" + 0.008*"gracias" +  
0.007*"senado" + 0.006*"concerta" + 0.005*"@saladehistoria" + 0.005*"vivo" + 0.005*"feliz"  
TOPIC: 4  
WORDS: 0.011*"educación" + 0.010*"@cbellolio" + 0.008*"@valenzuelalevi" + 0.007*"casa" +  
0.007*"@lboric" + 0.005*"facultad" + 0.005*"rector" + 0.005*"@donmatas" + 0.005*"arenas" +  
0.005*"punta"
```

Figure 8: Los tópicos están ordenados por coherencia y los números que aparecen al lado de cada palabra corresponde a la probabilidad de tal palabra en el tópico. Notar que sólo se muestran las 10 palabras con mayor probabilidad dentro del tópico, y dado el tamaño de  $V$  en este experimento, tales probabilidades son relativamente pequeñas. La coherencia promedio de los tópicos de esta imagen es  $-9.1$ .



### 2.3.2 Segunda iteración

Lo que corresponde ahora es optimizar la coherencia promedio con respecto a  $(\alpha, \eta)$  y `num_topics`. En la primera aproximación a esta tarea calculamos la `u_mass` sobre la grilla  $\{5, 6, 7, 8, 9, 10\} \times \{\text{auto}, 0.5\} \times \{\text{auto}, 0.5\}$  lo cual nos genera 24 puntos, ver Figura 9.

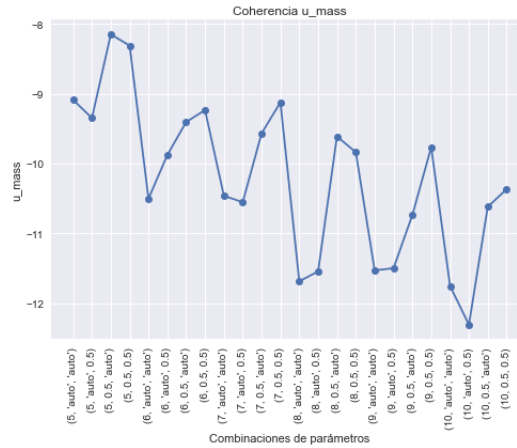


Figure 9: Coherencia por cada combinación de parámetros. Las tuplas del eje  $x$  son de la forma  $(\text{num\_topics}, \alpha, \eta)$ .

De la Figura 9 deducimos que la cantidad de tópicos escogidos es relevante y que mientras menos tópicos fijamos, más será la coherencia promedio. Es por esto que se decide fijar esta variable a 5 de aquí en adelante. Otra observación es que para número de tópicos fijos  $n$ , las mejores combinaciones son  $(n, 0.5, \text{auto})$  y  $(n, 0.5, 0.5)$ .

## References

- [1] *Latent Dirichlet Allocation*, Blei, Ng, Jordan, Journal of Machine Learning Research, 2003.
- [2] *Online Learning for Latent Dirichlet Allocation*, Matthew D. Hoffman, David M. Blei, Francis Bach, 2010.
- [3] *Exploring the Space of Topic Coherence Measures*, Michael Röder, Andreas Both, Alexander Hinneburg, 2015.