

# **Proyecto Disinformationpoliticslab 2021 - UNIR**

## **Metodología de análisis (v1.0)**

Iago Ocaranza Prado – Espossible S.L.

Elias Said-Hung – Universidad Internacional de la  
Rioja

Javier Martínez Torres – Universidad de Vigo

## 1. Entorno de trabajo

Durante el desarrollo de todo este análisis, se trabajará en un entorno *python*, utilizando ficheros tipo *notebook* de *Jupyter*, con un paradigma de programación exploratorio de scripting y fragmentos de código para ir generando los resultados requeridos. Este tipo de paradigmas son comúnmente utilizados en el ámbito académico y científico.

Como lenguaje de programación, se usará únicamente *python* como hemos mencionado, con una serie de dependencias relacionadas con NLP o *Natural language Processing*, así como librerías de análisis de datos tipo *pandas*, *numpy*, *scikit*, *sklearn* etc.

## 2. Dataset de origen

Partimos en este análisis de los siguientes *datasets*:

- *Dataset* de *Tweets* - Recopilación de *tweets* en excel con sus atributos principales
- *Dataset* de fake news provenientes de Maldita.es
- *Dataset* de fake news provenientes de Newtral
- **Preparación de los datos de origen**

Lo primero que realizamos en este estudio es una clasificación de *tweets*, para posteriormente poder identificarlos con su correspondiente actor político, perfil, ideología...etc

Para ello, leemos el fichero excel mediante la librería *pandas*, y lo convertimos en un *panda dataframe*.

En este punto, ya podemos empezar a trabajar sobre los datos de origen, haciendo una primera clasificación por:

- Tipo de actor político (líder o portavoz)

- Partido político
- Perfil ideológico
- Ámbito

Esta clasificación se realiza en base a la información recogida en el excel de identificación de actores políticos: “*Usuarios para OE1, OE2, OE3, OE4.xlsx*”

A mayores, realizamos una detección de idioma, debido a que ciertos análisis semánticos solo los podremos realizar sobre los *tweets* en idioma español. La librería utilizada para este análisis es *polyglot*

- **Tokenización**

Este proceso supone un primer pre-procesado de los *tweets*, que usaremos en los sucesivos análisis de este estudio como punto de partida.

Vamos a generar un nuevo *dataframe*, que consiste en un listado de palabras o *tokens* indexados por *tweet* para su posterior análisis.

Librería utilizada

**tmtoolkit**: Text mining and topic modeling toolkit

Para la tokenización, usamos el siguiente *snippet* de código:

```
corpus = tmtkCorpus(docs=doc_dict)
preproc = TMTPreproc(corpus, language='es')
preproc.pos_tag()
    .filter_for_pos('N')
    .clean_tokens(remove_numbers=True, remove_shorter_than=2, remove_longer_than=20)
    .tokens_to_lowercase()
    .remove_special_chars_in_tokens()
```

La librería extrae solamente los nombres, en minúscula, eliminando caracteres especiales, números y palabras con longitud menor que 2 y mayor que 20. Cada uno de estos elementos se considerará término o token durante todo el análisis.

### 3. Análisis de tópicos

El objetivo de este análisis, es tener una distribución de tópicos de cada uno de los actores políticos. Partimos del *dataset* de tokens de *tweets* previamente calculado.

- **Obtención del número óptimo de tópicos para un *corpus* dado**

En este apartado, lo que haremos será calcular mediante *tmtoolkit* una serie de métricas que nos indicarán el número óptimo de tópicos en los que dividir el *dataset* para que los términos estén mejor contextualizados. En particular, calcularemos las siguientes métricas para cada usuario de *tweeter*:

- *cao\_juan\_2009*
- *arun\_2010*
- *griffiths\_2004*
- *coherence\_mimno\_2011*

Generamos una gráfica ilustrativa para representar el valor de cada una de estas métricas, y visualmente escogeremos el valor más apropiado que satisfaga los criterios de cada una de ellas.

De esta manera, obtendremos una variable de configuración de número de tópicos óptimo para cada cuenta de *twitter*.

- **Cálculo de los términos de cada tópico en base al número óptimo de tópicos**

Con esta configuración previa, hacemos un análisis mediante el método *Latent Dirichlet Allocation (LDA)* de cada cuenta de *twitter*, generando así un fichero de resultados para cada una de ellas, que podemos encontrar en [https://figshare.com/articles/dataset/Análisis\\_de\\_tópicos\\_publicados\\_por\\_actores\\_políticos\\_en\\_tweets\\_en\\_Twitter/16435281](https://figshare.com/articles/dataset/Análisis_de_tópicos_publicados_por_actores_políticos_en_tweets_en_Twitter/16435281).

## 4. Categorización de contenido desinformativo

Partimos de un *dataset* de *tweets*. El objetivo de este apartado es determinar el grado de *desinformatividad* de cada uno de estos *tweets*.

- **Tokenizar la base de datos de fake news**

Utilizamos el mismo procedimiento que en el procesado inicial de *tweets*, pero en este caso sobre los *dataset* de noticias agregados para obtener un dataframe con el total de tokens de fake news indexados por noticia.

- **Comparativa cruzada de tokens de *tweet* con tokens de fake news**

De los dos *datasets* anteriores, tokens de *tweets* y de noticias, elaboramos una matriz producto resultante de igualar cada uno de los tokens elemento a elemento. Esta matriz resultante contiene las coincidencias de pares token *tweet* vs token noticia.

- **Nivel cuantitativo de desinformación de *tweet***

Para este paso, asumimos que, si un *tweet* proviene de un contenido desinformativo, éste se va a parecer mucho a una de entre todas las noticias, y poco o nada al resto. De esta manera, sobre la matriz global hacemos un *groupby* sobre el *tweet*, y nos quedamos con:

- máximo de cruces *tweet*/noticia para cada *tweet*
- Dividimos el número de cruces de cada *tweet* con cada noticia por la longitud de la misma, y calculamos también el máximo de este nuevo indicador
- Índice de noticia con máximo de cruces
- Índice de noticia con máximo de cruces normalizado por longitud

El hecho de generar ese segundo indicador normalizado, es que las noticias más largas tienen una

mayor probabilidad de generar coincidencias aleatorias con palabras del *tweet*, pero eso no significa que sean más parecidos y debemos considerar el *tweet* más desinformativo.

De todas formas, mantendremos ambos indicadores y usamos los dos indistintamente según podamos aislar este fenómeno y no nos repercuta en nuestro análisis

A continuación, generamos los indicadores cualitativos correspondientes a:

- Cruces de noticias -> nivel desinformativo
- Cruces de noticias normalizadas por longitud -> nivel desinformativo normalizado

Estas variables cualitativas se generan mediante la división en cuantiles del *dataset*, en nuestro caso 5: *very low, low, medium, high, very high*.

Estos indicadores cualitativos nos indican qué *tweet* tiene más coincidencia desinformativa y con cuál de los bulos.

- **Nivel cualitativo de desinformación de *tweet***

Se decide en este paso, clasificar en 5 niveles discretos el valor de desinformación que transmite cada *tweet*. Para ello, utilizamos la función `qcut` de Numpy, de tal forma que distribuimos este vector en quintiles, asociando a cada *tweet* el nivel discreto que le corresponde de contenido desinformativo: *Muy bajo, bajo, medio, alto, muy alto*.

Generando los siguientes indicadores cualitativos:

- Cruces de noticias -> nivel desinformativo
- Cruces de noticias normalizadas por longitud -> nivel desinformativo normalizado

- **Indicador de *virilidad***

Hemos generado también un indicador que representa hasta dónde se ha transmitido el *tweet* y bien que ha sido recibido. Para ello, hemos seguido la expresión:

$retweet\_count + favorite\_count / 2$

Lo hemos aplicado al *dataset* de *tweets*, y generamos la correspondiente columna en el *dataframe* para poder usarla en otros análisis.

## 5. *Hate speech*

El objetivo de este análisis, es calcular cualitativamente cuánto lenguaje de odio tienen cada uno de los *tweets*, para extrapolarlo con otro tipo de variables como el actor político, o el ámbito, por ejemplo.

- **Tokenización y lematización**

En este análisis, partimos del *dataframe* anterior de *tweets* pre-filtrados en español. En este caso, hemos decidido tanto tokenizar como lematizar. Este último proceso consiste en llevar cada uno de los tokens (nombres) a su término raíz (lema). De esta forma, tendremos más posibilidades de detectar el lenguaje de odio en el siguiente paso, debido al tipo de análisis que realizamos. Utilizamos el mismo *snippet* de código que para la tokenización ya que también obtenemos el lemma mediante esa función

- **Análisis sentimental**

En este apartado, analizamos la polarización sentimental de los mensajes. Para ello, construiremos un indicador sentimental en el intervalo  $[-1, 1]$  que nos indica cuánto de negativo o positivo es cada uno de los mensajes en particular.

Utilizamos la librería <http://sentistrength.wlv.ac.uk/> la cual se basa en unos modelos dediccionario preestablecidos.

Localizamos un modelo publicado por parte de Felipe Bravo como parte de un trabajo, que

podemos obtener mediante la plataforma open source Github:

<https://github.com/felipebravom/StaticTwitterSent/tree/master/extra/SentiStrength/spanish>

Aplicamos a cada mensaje el análisis *sentistrength* que nos devuelve un indicador independiente de positividad y otro de negatividad sobre cada mensaje. Agregamos ambos para conseguir el indicador sentimental único deseado

```
sent_map = {
    -4: 'negative',
    -3: 'negative',
    -2: 'negative',
    -1: 'neutral',
    0: 'neutral',
    1: 'neutral',
    2: 'positive',
    3: 'positive',
    4: 'positive'
}
```

- **Análisis del discurso del odio: Inmigración, insultos, xenofobia, misoginia**

En este apartado analizamos cuánto mensaje de odio tienen los mensajes, en base a los siguientes indicadores:

- Discurso del odio contra inmigración
- Insultos
- Xenofobia
- Misoginia

Utilizamos para ello, el siguiente repositorio:

<https://github.com/fmplaza/hate-speech-spanish-lexicons>



Dónde podemos encontrar un *dataset* de palabras relacionadas para uno de estos 4 aspectos del *hate speech*.

Utilizamos la tokenización y la lematización, para cruzar cada mensaje con cada uno de estos *datasets*, generando un indicador del número de coincidencias de tokens/lemas sobre cada uno de los *datasets*.

Ya tenemos un indicador cuantitativo del *hate speech*. Para obtener el indicador cualitativo, hacemos una división por cuantiles (3) para categorizar los *tweets* en base a 3 niveles por indicador: bajo, medio alto

- **Volcado de datos**

Finalmente, guardamos el *dataset* completo, con todas las nuevas columnas que hemos ido calculando en los apartados 2 y 3 en [https://figshare.com/articles/dataset/Datos\\_procesados\\_proyecto\\_RETOS\\_final/16434885](https://figshare.com/articles/dataset/Datos_procesados_proyecto_RETOS_final/16434885).

## 6. Encontrar relaciones de dependencia entre variables

En este análisis, evaluaremos la posibilidad de que exista dependencia entre las variables asociadas a los *tweets* / actores políticos mediante la metodología estadística del chi cuadrado.

Planteamos la hipótesis nula, y calculamos tanto los valores esperados como el valor de chi cuadrado resultante.

Este paso se realiza en excel manualmente, mediante el uso de tablas dinámicas contra la primera hora de resultados del fichero de *tweets\_procesados.xls* del apartado anterior.

Se añade una hoja por cada uno de los correspondientes análisis:

- Nivel desinformativo vs tipo de mensaje (*retweet* vs no *retweet*)
- Nivel desinformativo vs factor de replicabilidad
- Nivel desinformativo vs perfil
- Nivel desinformativo vs ámbito

A su vez, se calcula también el coeficiente de correlación, así como Crámers V., para obtener más indicadores estadísticos de dependencia

Los resultados se vuelcan en el excel de salida en [https://figshare.com/articles/dataset/Datos\\_procesados\\_proyecto\\_RETOS\\_final/16434885](https://figshare.com/articles/dataset/Datos_procesados_proyecto_RETOS_final/16434885).