Benjamin Gutierrez Garcia bencouver@gmail.com

Topic Modelling
Latent Dirichlet Allocation para el “all the news”dataset

Introduction

Modelación de temas es un tema de gran interés para aquellos que estudian las redes sociales y la miríada de acervos bibliográficos y documentales con los que contamos. Mientras más material es digitalizado y producido en tiempo real por los medios de comunicación, más importancia adquieren las técnicas de aprendizaje de máquina dedicadas a buscar patrones y tendencias.

Un tema muy importante es el de la búsqueda de temas y sentimiento en los medios noticiosos de comunicación. En este documento describimos la implementación de un código que mina temas en el dataset “all the news” [1], que contiene información de artículos publicados en diversos medios de comunicación de habla inglesa.

Vamos a implementar una técnica conocida como Distribución latente de Dirichlet (*Latent Dirichlet Allocation, LDA*[2]*),* ya que es muy popular*.* LDa asume que cualquier documento (el encabezado de una nota en la web, por ejemplo, o el cuerpo de la misma) contienen una mezcla o combinación de temas. Esos temas generan palabras basados en una distribución de probabilidad. Dado un dataset de documentos --como el archivo articles1.csv en all-the-news-- LDA trata de reconstruir qué temas producirían estos documentos. LDA traduce el problema a factorizar matrices., ya que en cierto espacio vectorial, cualquier *corpus* (o colección de documentos) puede representarse como una matriz. No voy a entrar en detalles pero esta es básicamente la idea detras de LDA[3].

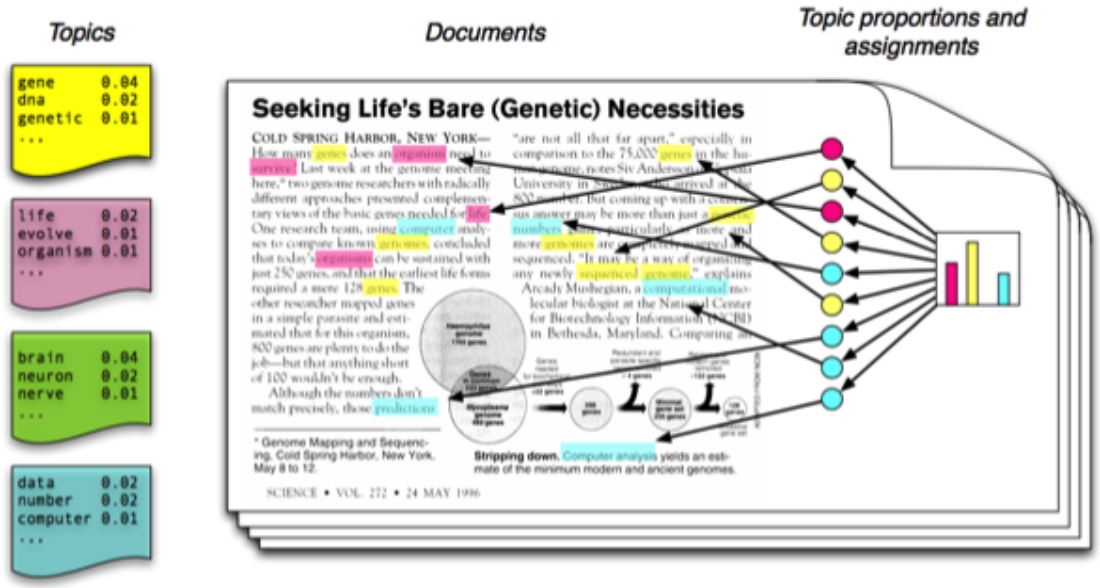


Figure source: Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

Implementación

Este código es relativamente sencillo, utiliza las bibliotecas natural language toolkit, NLTK, y Gensim. El dataset all-the-news debe obtenerse manualmente y colocarse en un directorio llamada data, debajo de el directorio de trabajo donde se despliegue nuestro código (`topic_modeling.py`). No lo incluyo en el repositorio porque el tamaño del mismo. De hecho solo utilice el archivo articles1.csv, com 50,000 líneas. Este dataset tiene el esquema:

```
[[columnasinheader]],id,title,publication,author,date,year,month,url,content
```

Solo vamos a utilizar el título en este código. Los datos son leídos con pandas, teniendo cuidado de eliminar líneas con formatos defectuosos (Tambien experimente con el dataset de un millon de lineas de ABC News, y en el abundan líneas sucias).

La primera acción es eliminar las palabras conocidas como *stopwords* (palabras irrelevantes), y tambien se transforman algunos tipos de datos en flotantes para agilizar las operaciones de iteracion, en forma similar a como datos categóricos se manejaron en el problema de CIFAR100. A continuación se crea un objeto para el modelo de LDA y se entrena en la matriz de documentos que se forma con los datos. Gensim permite llevar a cabo este entrenamiento a partir de un *training corpus*, y posteriormente a realizar la inferencia de la distribución de temas en documentos no vistos por el modelo anteriormente.

Resultados e interpretación

El código finalmente arroja las combinaciones lineales de palabras más relevantes para cada tema con sus pesos o probabilidades, una línea por tema. Tenemos diez

temas con diez palabras principales en esta implementación.

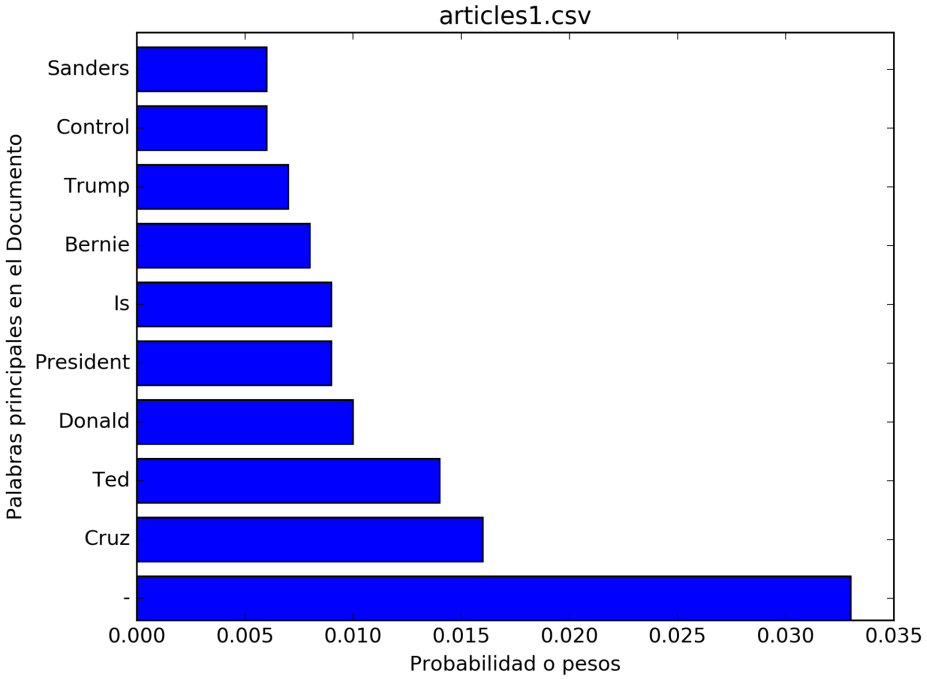
Este código se ejecutó en Ubuntu 16.04.4 LTS, python2.7.12. En una máquina de 8 cores (Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz) con 32 GB de RAM. El procesar el archivo articles1.csv, con 50,000 líneas. toma aproximadamente del orden de una hora.

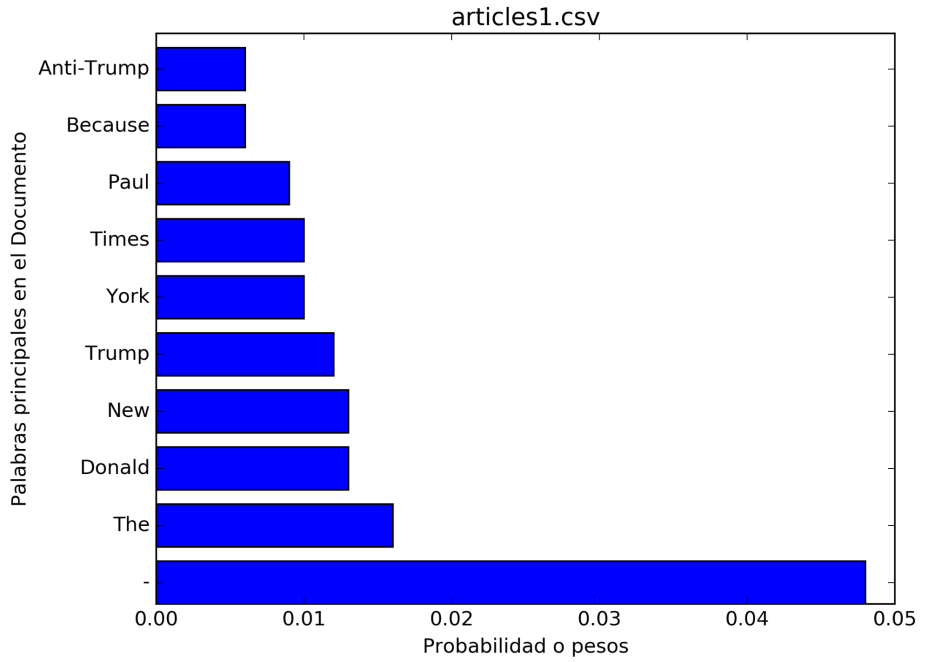
"Raw" output del código:

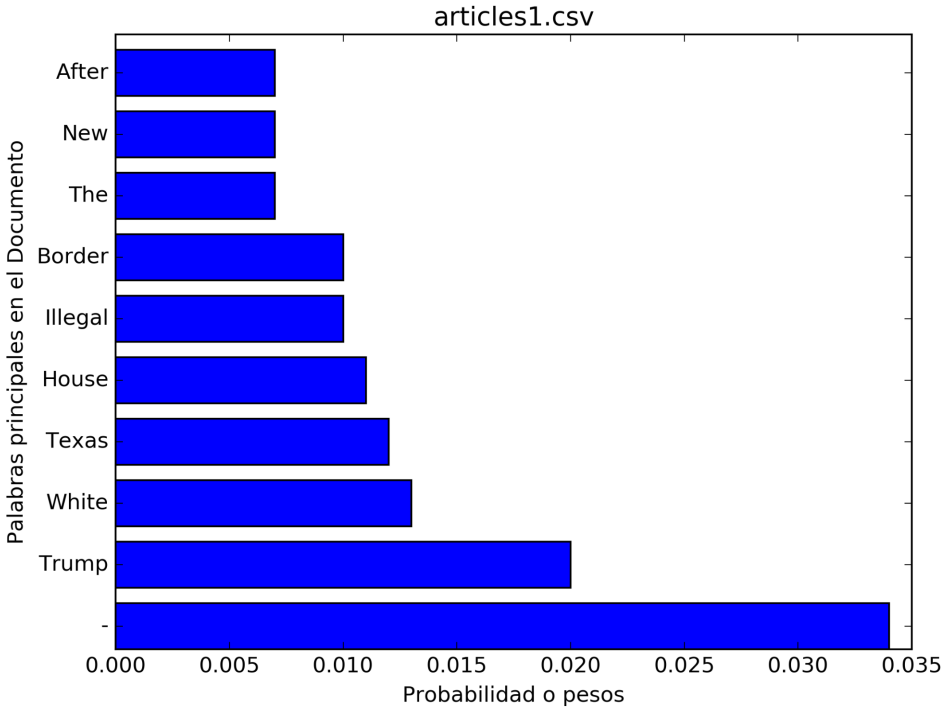
```
benjamin@higgs:~/topic$ python topic_modeling.py
c = 49999 / 49999
0.048*"-" + 0.016*"The" + 0.013*"Donald" + 0.013*"New" + 0.012*"Trump" +
0.010*"York" + 0.010*"Times" + 0.009*"Paul" + 0.006*"Because" + 0.006*"Anti-
Trump"
0.034*"-" + 0.020*"Trump" + 0.013*"White" + 0.012*"Texas" + 0.011*"House" +
0.010*"Illegal" + 0.010*"Border" + 0.007*"The" + 0.007*"New" + 0.007*"After"
0.044*"-" + 0.037*"Hillary" + 0.036*"Clinton" + 0.014*"Trump" + 0.010*"The" +
0.010*"New" + 0.007*"Times" + 0.007*"York" + 0.007*"Donald" +
0.005*"Clinton's"
0.033*"-" + 0.016*"Cruz" + 0.014*"Ted" + 0.010*"Donald" + 0.009*"President" +
0.009*"Is" + 0.008*"Bernie" + 0.007*"Trump" + 0.006*"Control" +
0.006*"Sanders"
0.032*"-" + 0.020*"Islamic" + 0.018*"Trump" + 0.017*"State" + 0.010*"American"
+ 0.009*"John" + 0.008*"To" + 0.007*"The" + 0.006*"New" + 0.006*"Court"
0.038*"-" + 0.012*"Trump" + 0.010*"The" + 0.009*"New" + 0.007*"Rubio" +
0.007*"York" + 0.007*"Ryan" + 0.006*"Open" + 0.006*"Times" + 0.006*"Is"
0.192*"Breitbart" + 0.102*"-" + 0.021*"Trump" + 0.006*"Is" + 0.005*"Not" +
0.005*"Obama" + 0.005*"Will" + 0.005*"Trump:" + 0.005*"The" + 0.004*"GOP"
0.036*"To" + 0.026*"-" + 0.011*"Man" + 0.010*"Migrant" + 0.007*"Terror" +
0.007*"The" + 0.006*"New" + 0.006*"Following" + 0.006*"Mexican" +
0.006*"Police"
0.031*"-" + 0.014*"We" + 0.013*"Watch:" + 0.008*"After" + 0.007*"The" +
0.006*"New" + 0.006*"WATCH:" + 0.005*"Attack" + 0.005*"Two" +
0.005*"Terrorist"
0.035*"-" + 0.009*"Gun" + 0.008*"The" + 0.008*"New" + 0.006*"Times" +
0.006*"Must" + 0.006*"York" + 0.005*"Email" + 0.005*"Trump" + 0.005*"Days"
```

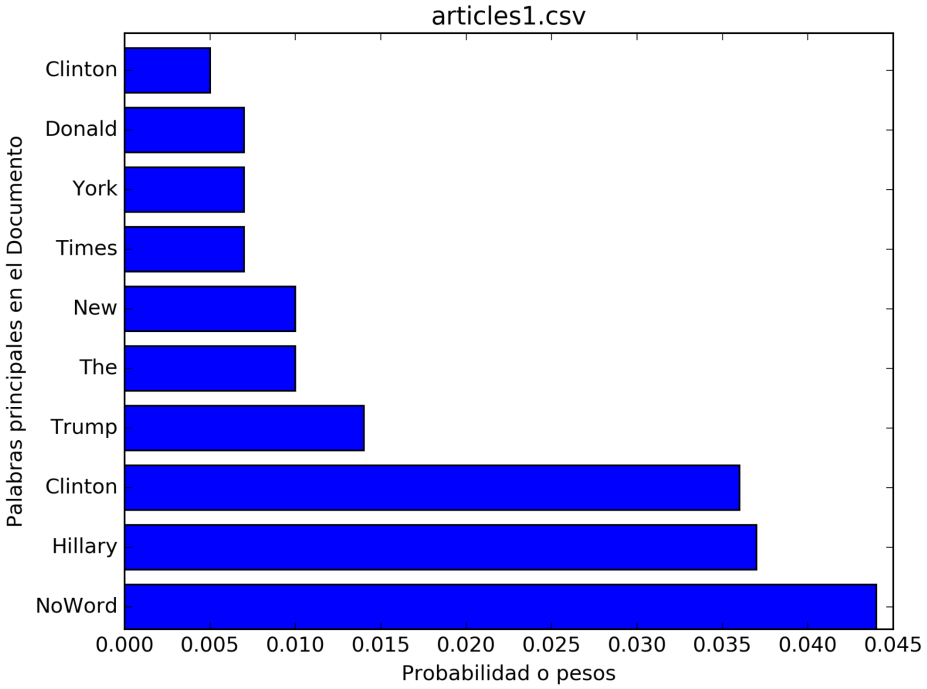
Estos datos de salida se encuentran en el archivo `example.txt`.

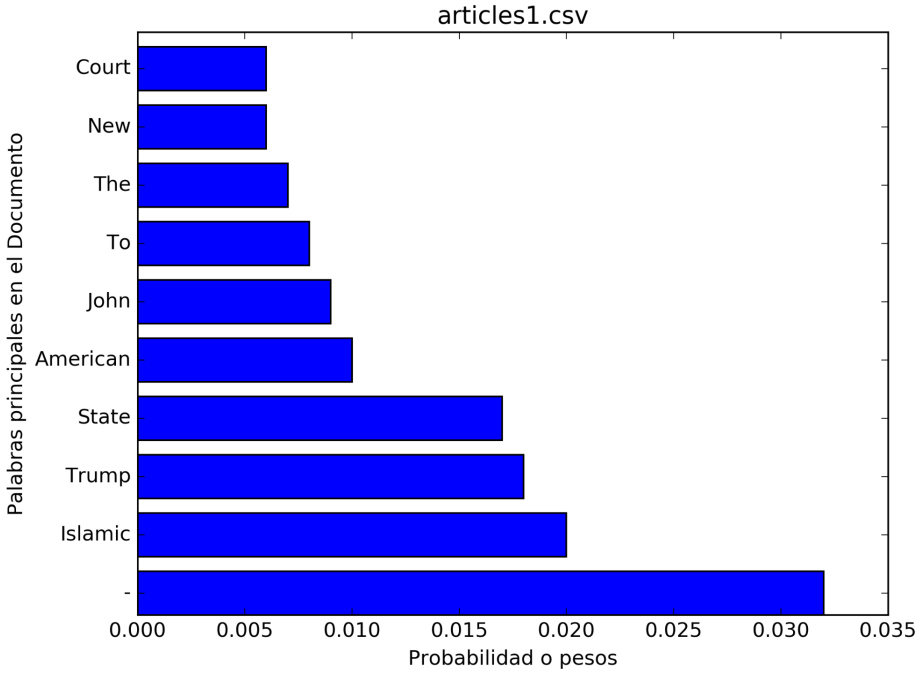
Hay muchas formas de visualizar esta información de forma más ilustrativa, y existen muchísimas bibliotecas de software de alta calidad en python y R para tal propósito. Decido presentar un histograma para algunos de los temas, de donde es mas facil inferir cuales es el tema dominante. Se incluye dos scrips auxiliares, `tocsv.sh` y `plot_topic_histogram.py` para generar los histogramas siguientes (cada uno corresponde a un *Topic* o tema distinto):

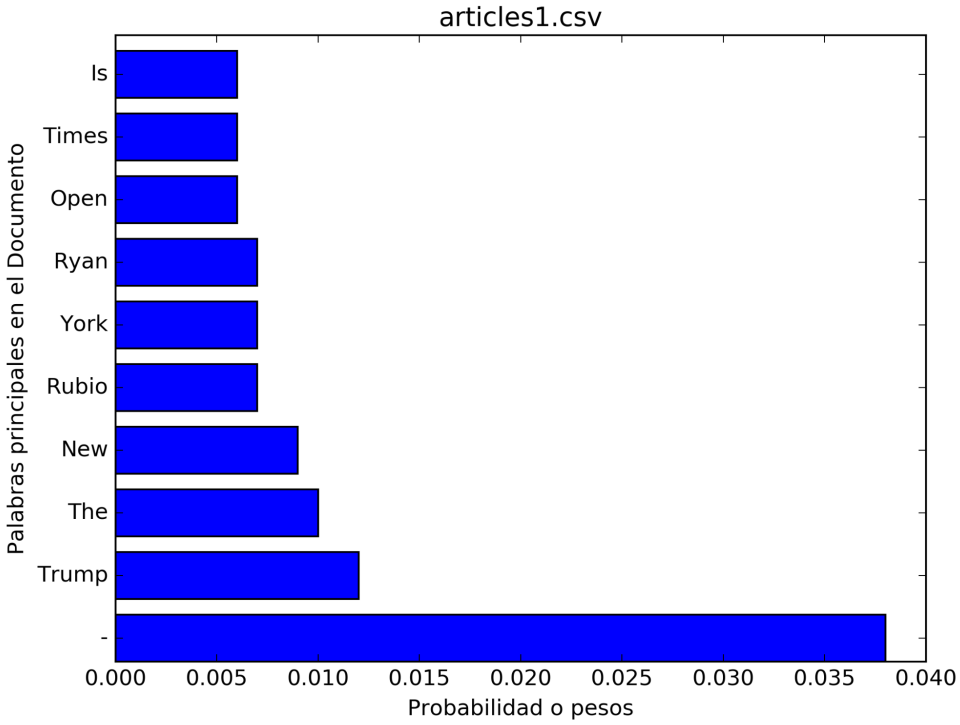

articles1.csv
Sanders
Control
Trump
Bernie
Is
President
Donald
Ted
Cruz
-
Palabras principales en el Documento
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
Probabilidad o pesos

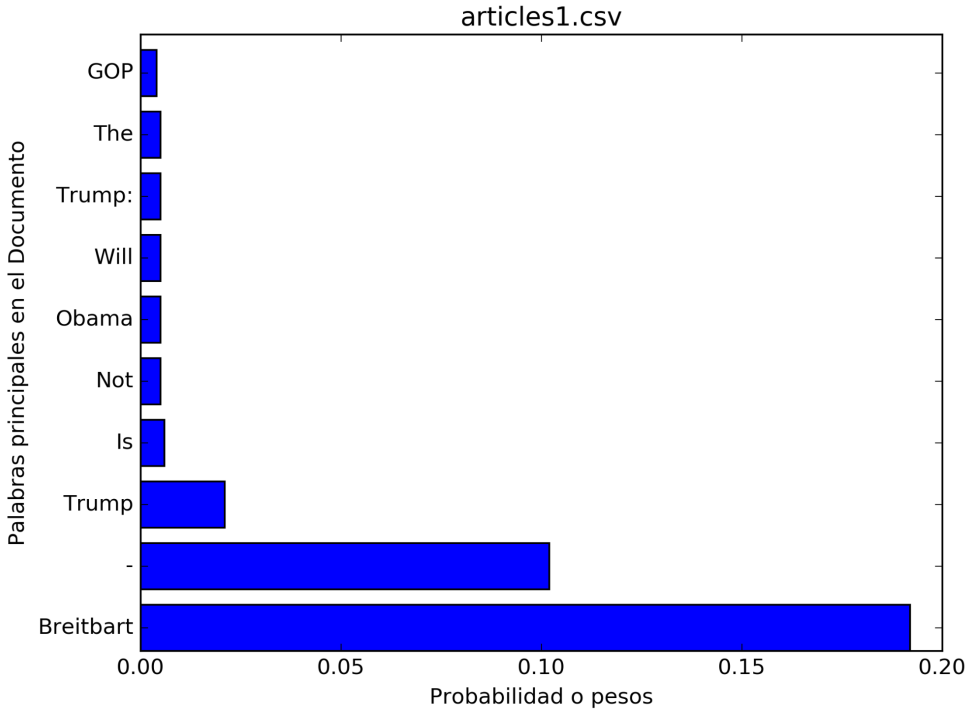

articles1.csv
Anti-Trump
Because
Paul
Times
York
Trump
New
Donald
The
-
Palabras principales en el Documento
0.00
0.01
0.02
0.03
0.04
0.05
Probabilidad o pesos

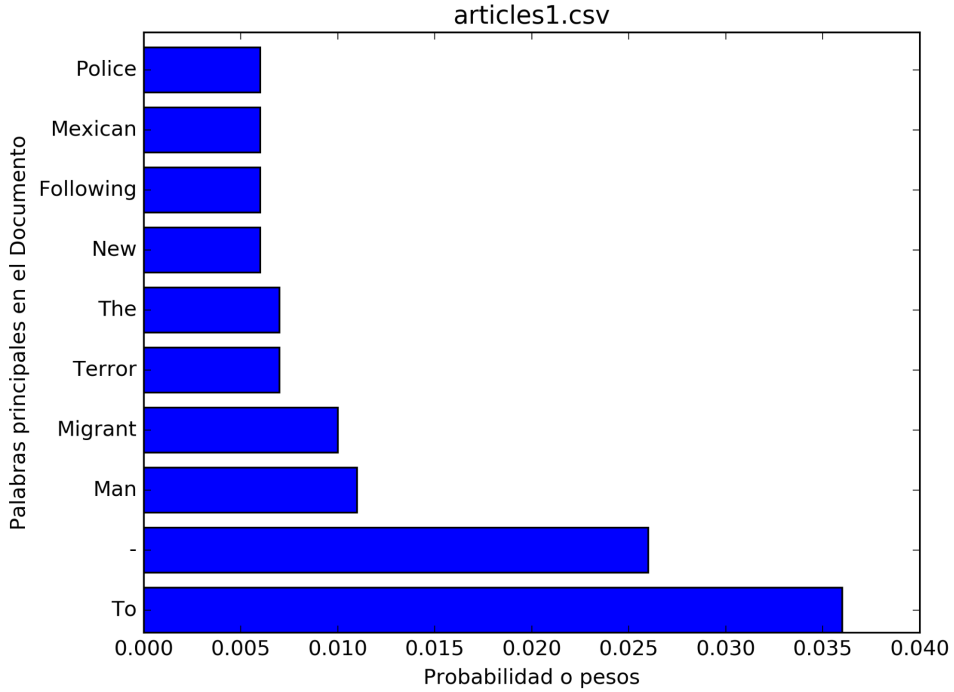

articles1.csv
After
New
The
Border
Illegal
House
Texas
White
Trump
-
Palabras principales en el Documento
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
Probabilidad o pesos

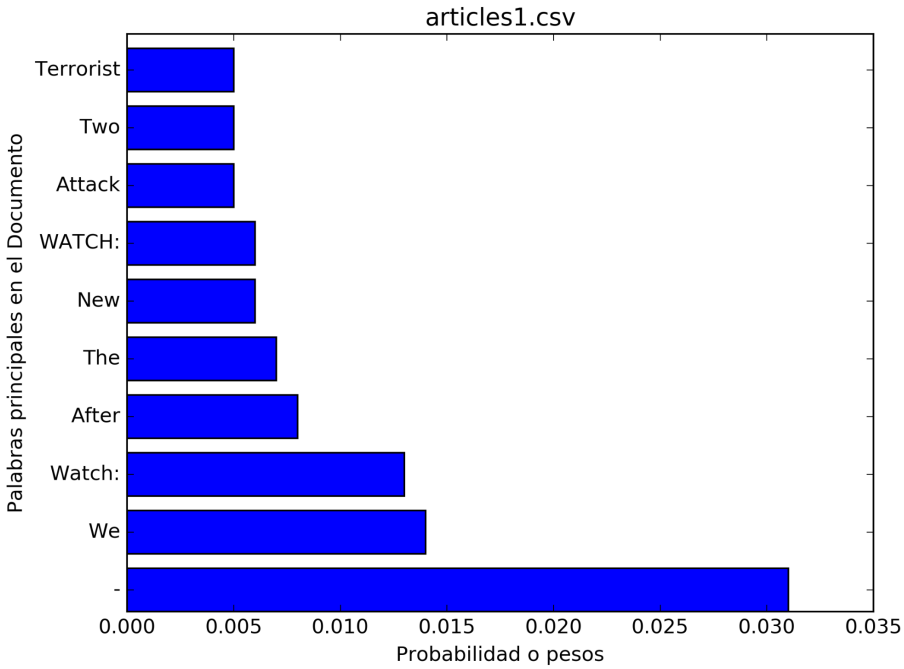

articles1.csv
Clinton
Donald
York
Times
New
The
Trump
Clinton
Hillary
NoWord
Palabras principales en el Documento
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
0.045
Probabilidad o pesos

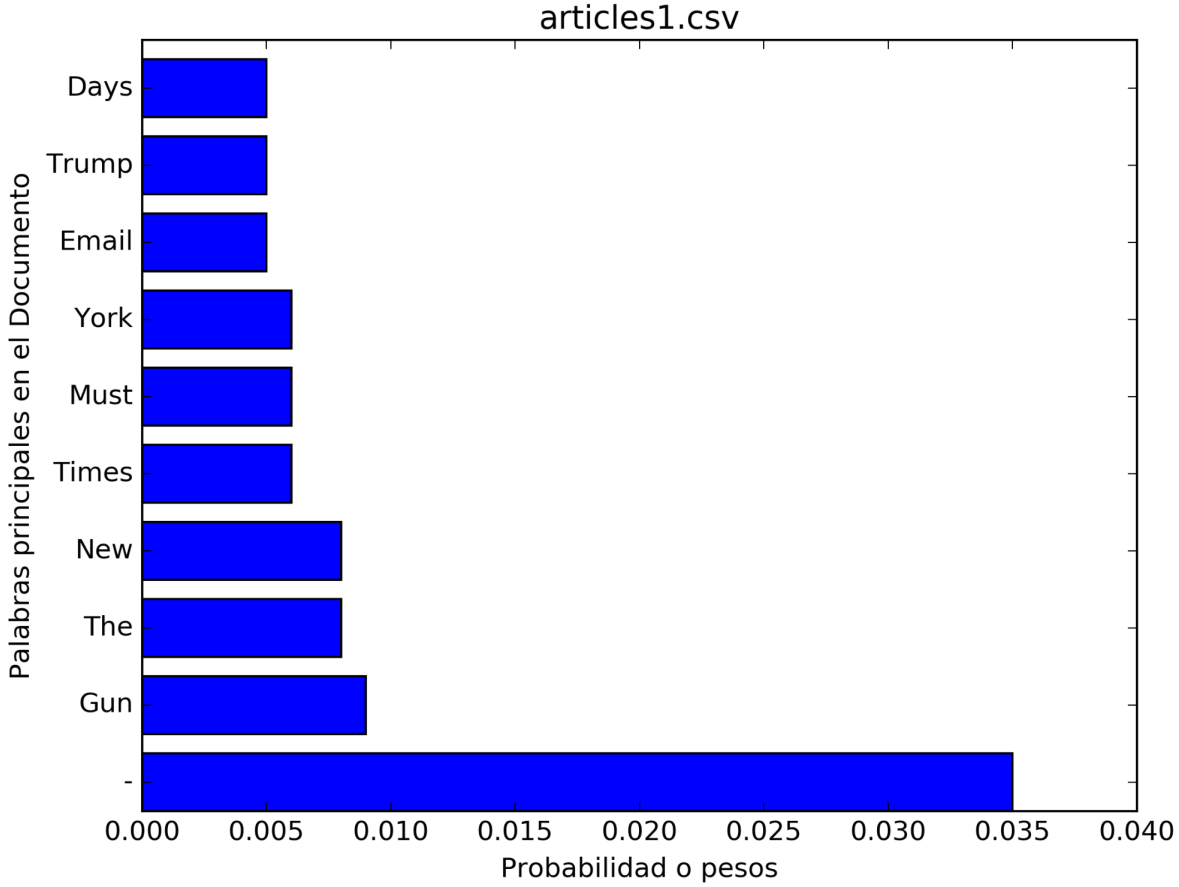

articles1.csv
Palabras principales en el Documento
Court
New
The
To
John
American
State
Trump
Islamic
-
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
Probabilidad o pesos


articles1.csv
Palabras principales en el Documento
Is
Times
Open
Ryan
York
Rubio
New
The
Trump
-
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
Probabilidad o pesos


articles1.csv
Palabras principales en el Documento
GOP
The
Trump:
Will
Obama
Not
Is
Trump
-
Breitbart
0.00
0.05
0.10
0.15
0.20
Probabilidad o pesos


articles1.csv
Palabras principales en el Documento
Police
Mexican
Following
New
The
Terror
Migrant
Man
-
To
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
Probabilidad o pesos





Dado que el dataset corresponde a 2015-2016, los componentes mas fuertes reflejan las tendencias de noticias que mas capturaron a los medios, y que aun continuan “en el aire”. Un tema en verdad interesante, *no pun intended*.

---

[1] https://www.kaggle.com/snapcrack/all-the-news

[2] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I (January 2003). Lafferty, John, ed. "Latent Dirichlet Allocation". Journal of Machine Learning Research. 3 (4–5): pp. 993–1022.
[3] Arun R., Suresh V., Veni Madhavan C.E., Narasimha Murthy M.N. (2010) On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations. In: Zaki M.J., Yu J.X., Ravindran B., Pudi V. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2010. Lecture Notes in Computer Science, vol 6118. Springer, Berlin, Heidelberg