Universitat de València

Análisis de señales

Análisis semántico latente

Trabajo de la asignatura

Autores:

Carlos Martínez Sotorres
José Manuel Sánchez Aquilué

noviembre, 2021

Escuela Técnica Superior de Ingeniería (ETSE-UV)

Índice

1. Introducción **2**

2. Extracción de datos **2**

3. Exploración de datos **3**

4. Preprocesado de datos **3**

5. Bolsa de palabras **4**

6. Modelo LSA **5**

7. Resultados **7**

8. Conclusiones **9**

1. Introducción

La clasificación de documentos por tema o topic modeling es algo que cada vez cobra mayor importancia en nuestro mundo moderno. Desde poder ayudarnos a encontrar mejores resultados en Google o cualquier otro motor de búsqueda, obtener mejores recomendaciones en nuestra navegación por Internet o incluso hasta en la selección de candidatos para un puesto de trabajo, la clasificación documentos tiene o empieza a tener un papel bastante significativo. Es por ello que en este trabajo intentaremos explorarlo superficialmente, en concreto, el análisis semántico por Latent Semantic Analysis (LSA).

Esta técnica está basada en descomposición en valores singulares vista en clase. El objetivo de este trabajo es identificar patrones en una colección de artículos. Para ello será necesario entender el conjunto con el que estamos trabajando, el modelo de bolsa de palabras, LSA y la coherencia. A parte de procesar el input adecuadamente hasta dar con la matriz de término-documentos y ajustar el modelo obtener los mejores resultados.

A la hora de llevar a cabo la implementación nos hemos apoyado en las funciones de la biblioteca gensim, que tiene definidas funciones de Topic model y NLP, que permiten implementar este tipo de modelos sin necesidad de programar los detalles a bajo nivel. El código fuente se encuentra disponible en: `https://colab.research.google.com/drive/17pHElZQf3UNwdVQkScNGVqHR4XQFSJxu?usp=sharing` [Último acceso 12 de Noviembre de 2021]

2. Extracción de datos

Para la construcción de nuestro modelo hace falta un conjunto de documentos de cierto volumen. Hemos tomado la decisión de utilizar el conjunto de datos de artículos disponible en datacamp [2], el cual cuenta con 4551 documentos.

En materia de importación, consideramos que lo más eficaz es realizar la descarga del conjunto online (por medio de peticiones HTTP) cada vez que ejecutemos el código. De este modo, contaremos siempre con la versión más actualizada del fichero.

El conjunto de datos se encuentra almacenado en un fichero de texto plano de extensión txt dónde cada artículo está presente en una línea. Cabe aclarar que la codificación original del fichero no es válida para empezar trabajar directamente, hace falta hacer una decodificación previa a utf-8.

3. Exploración de datos

Como se ha mencionado anteriormente, nuestra colección consta de 4551 documentos, de una media de 5478 palabras. Ahora bien, la distribución de la extensión de los textos no es en absoluto uniforme (Tabla 1). Por lo general son artículos de entre 2500 y 60000 caracteres, aunque hay algunos artículos cuya extensión supera con creces la media. En la figura 2 se puede visualizar la distribución.

media	5477.65
std	6357.94
min	410
25 %	2726
50 %	4321
75 %	5962.5
max	96917

Tabla 1: Distribución del número de caracteres de los textos.

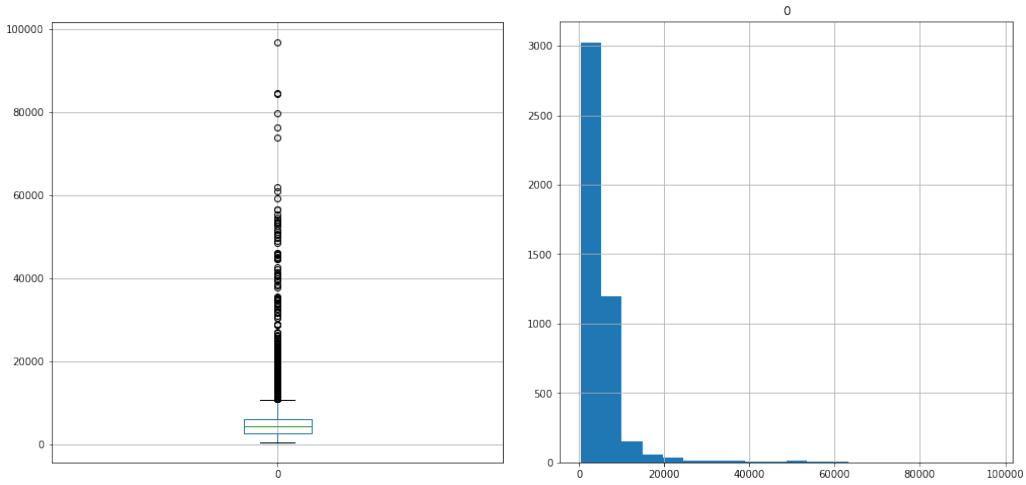

Figura 2: Boxplot (izq) e histograma (der) de la distribución del número de caracteres de los documentos.

Por otro lado, tras deshacernos de las stopwords y hacer stemming, hemos construido un diccionario en base a estos artículos de 49792 palabras.

4. Preprocesado de datos

Para preparar la colección de datos se realiza un preprocesado de datos o Data Cleaning, que consiste en corregir y normalizar los documentos para proporcionar una entrada de datos válida al algoritmo. Se realiza mediante los siguientes pasos:

1. La conversión a minúsculas de todo el texto. Debido a que una misma palabra

será tomada como diferente por el algoritmo si no coincide completamente, por lo que una diferencia en una letra mayúscula altera el resultado.

2. Tokenizar mediante espacios en blanco. Esto implica dividir y trocear los documentos por palabras completas, tokens, que serán los datos de entrada del algoritmo.

3. Eliminar "stopwords". Las "stopwords" son palabras que en un lenguaje determinado se considera que no aportan significado al texto y que se usan con fines de cohesión y estructura léxica, como pueden ser las preposiciones, por lo que se eliminan de la colección de datos para mantener tan sólo las palabras con un aporte semántico.

4. Stemming. El stemming consiste en derivar las palabras para obtener su raíz y reducirlas a esta, con lo cual se decrementa el número de palabras diferentes que aparecen que contengan un significado similar, como podría ser el caso de "leave" y "leaving" manteniendo el número de veces que aparece el morfema. Aplicando stemming se reducen ambas a la raíz, "leav" y se reduce de dos a una la cantidad de palabras diferentes que procesa el algoritmo.

5. Bolsa de palabras

El proceso de análisis semántico que seguiremos empleando el modelo de Latent Semantic Analysis (LSA) hará uso de una bolsa de palabras (bag of words). El modelo de bolsa de palabras es una forma de representar datos de texto cuando se modeliza texto con algoritmos de aprendizaje automático. Para entender este modelo primero debemos entender algunos de los problemas que nos surgen al tratar con texto.

Cuando tratamos texto con algoritmos de aprendizaje automático encontramos varios obstáculos ya que estos tienen preferencia por vectores de longitud predeterminada. De hecho, estos algoritmos no pueden tratar con texto en sí y por tanto este debe de ser reconvertido a vectores numéricos.

"En el procesamiento del lenguaje, los vectores x se derivan de datos textuales, con el fin de reflejar varias propiedades lingüísticas del texto." - Page 65, Neural Network Methods in Natural Language Processing, 2017. [3]

Esto se denomina extracción de características y uno de los modelos más populares es el de bolsa de palabras, al que a partir de ahora nos referiremos como BoW.

BoW es simplemente una representación de las características del texto que tiene en cuenta principalmente dos cosas:

1. Un vocabulario de palabras conocidas.

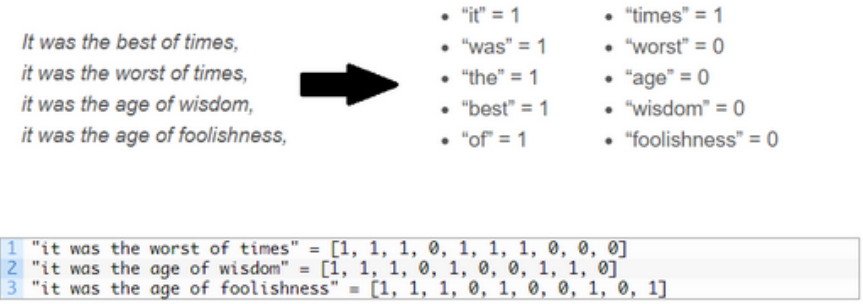

Figura 3: Ejemplo de codificación de bolsa de palabras.

2. La ocurrencia de dichas palabras en el texto.

Un aspecto destacable de esto es que es importante ver que al BoW le es indiferente el orden ni la posición de las palabras en en el texto, solo le interesa cuántas veces aparece en el texto. El grado de compleja que pueda llegar a ser una bolsa de palabras pasa por muchos niveles según como se quiera definir el diccionario de palabras (tokens) y cómo se quiera puntuar su aparición.

Los pasos a tener en cuenta en una bolsa de palabras son los siguientes:

1. Recolección de datos.
2. Diseño del vocabulario.
3. Creación de los vectores de documento.

En la figura 3 podemos ver un ejemplo de cómo se codificarían las palabras en vectores numéricos para poder ser interpretadas por los algoritmos de aprendizaje automático.

6. Modelo LSA

El modelo LSA (Latent Semantic Analysis) se apoya en un modelo de bolsa de palabras, lo que da como resultado una matriz de documentos de términos (número de apariciones de términos en un documento). Las filas representan términos y las columnas representan documentos. LSA va aprendiendo temas latentes utilizando la descomposición matricial de valores singulares en la matriz documento-término (Figura 4). LSA se usa habitualmente como una técnica de reducción de dimensión o reducción de ruido.

Recordemos la descomposición por valores singulares:

$$M = U\Sigma V^*$$

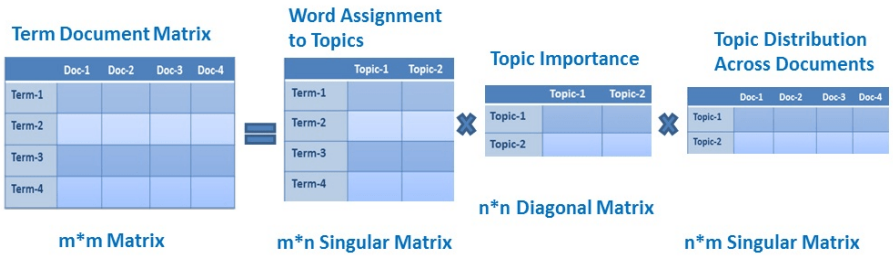

Figura 4: Descomposición de la matriz documentos-termino. (Imagen extraída de datacamp).

Donde:

- M es una matriz mxm, en nuestro caso la matriz documento-término.
- U es una matriz singular (determinante nulo) de tamaño mxn
- Σ es una matriz diagonal $n \times n$ con elementos reales positivos.
- V es una matriz singular $m \times n$, cuya traspuesta es V^* de dimensiones $n \times m$.

Otro detalle a tratar es la estimación del número óptimo "k" de topics. Una forma de determinar el número óptimo de temas es considerar cada tema como un grupo y averiguar la efectividad de un grupo utilizando el coeficiente de Silhouette.

Otra forma, y la que hemos utilizado más adelante en los experimentos, sería la medida de la coherencia del topic [6] [7]. La medida de coherencia dl topic es una métrica ampliamente utilizada para evaluar modelos de temas (análisis semántico). Cada tema generado tiene una lista de palabras. En la medida de coherencia del tema, encontrará la media / mediana de las puntuaciones de similitud de palabras por pares de las palabras de un tema. El alto valor del modelo de puntuación de coherencia de tema se considerará como un buen modelo de tema.

Si entramos más en detalle, en este ejemplo hemos empleado la medida de coherencia c_v, que se basa en una ventana deslizante, segmentación de un conjunto de las palabras top de ese tema y una medida de confirmación indirecta que utiliza información mutua puntual normalizada (NPMI) y la distancia del coseno. La fórmula resultante sería la siguiente:

$$\overrightarrow{v}(W') = \{ \sum_{w_i \in W'} NPMI(w_i, w_j)^{\gamma} \}_{j=1,\ldots,|W|} \tag{1}$$

$$NPMI(w_i, w_j)^{\gamma} = \left(\frac{\log \frac{P(w_i,w_j)+\epsilon}{P(w_i,w_j)}}{-\log(P(w_i, w_j) + \epsilon)}\right) \tag{2}$$

$$\phi_{S_i}(\overrightarrow{u}, \overrightarrow{w}) = \frac{\sum_{i=1}^{|W|} u_i \cdot w_i}{||\overrightarrow{u}||_2 \cdot ||\overrightarrow{w}||_2} \tag{3}$$

En materia de implementación, la biblioteca gensim tiene una clase llamada Dictionary que, si le pasas como argumento los documentos preprocesados, genera un diccionario. Asimismo esa clase tiene un atributo que dado un documento te devuelve su representación en bolsa de palabras. Así que aplicando esa función a todos los documentos obtendremos la matriz término-documento. Esa misma biblioteca tiene una clase llamada LsiModel que crea el modelo LSA a partir de la matriz anterior, el diccionario y el número de temas. Finalmente, también existe un modelo de coherencia ya definido, que se vale del modelo anterior, los textos y el diccionario.

En conclusión, el algoritmo LSA es el método más simple, fácil de entender e implementar. Es más rápido en comparación con otros algoritmos disponibles porque solo implica la descomposición de la matriz de términos del documento. Sin embargo también hemos de tener en cuenta el rango de la matriz, que limita el problema. Tampoco puede tener en cuenta distintos significados de una misma palabra y hay modelos que son más eficaces como el LDA (Latent Dirichlet allocation).

7. Resultados

Como hemos visto en el apartado anterior el número de temas es un hiperparámetro del modelo. En esta sección nos encargaremos de encontrar el modelo óptimo, es decir aquel que devuelva la mejor puntuación en función del número de temas.

Antes de empezar vamos a construir un modelo de siete temas y a pedirle que nos devuelva las 12 palabras más importantes de cada uno. En base a esas palabras determinaremos los temas.

- Tema 1: Elecciones presidenciales de EEUU de 2016 (trump, say, said, would, clinton, people, one, campaign, year, time...)
- Tema 2: Premier League (citi, v, h, 2016, 2017, unit, manchest, apr ,dec...)
- Tema 3: Brexit y elecciones en EEUU (trump, clinton, eu, say, would, donald, leav, uk,republican, cameron...)
- Tema 4: Futbol o Deportes en general (min, eu, goal, ball, play, said, say, leagu, leav, game...)
- Tema 5: Economía y Brexit (bank, eu, min, year, leav, cameron, market, rate, vote, say...)
- Tema 6: Economía (bank, say, peopl, trump, 1, min, eu, market, like...)
- Tema 7: Elecciones presidenciales de EEUU y presupuestos (say, min,vote, govern, poll, tax, statement, bank, budget, one...)

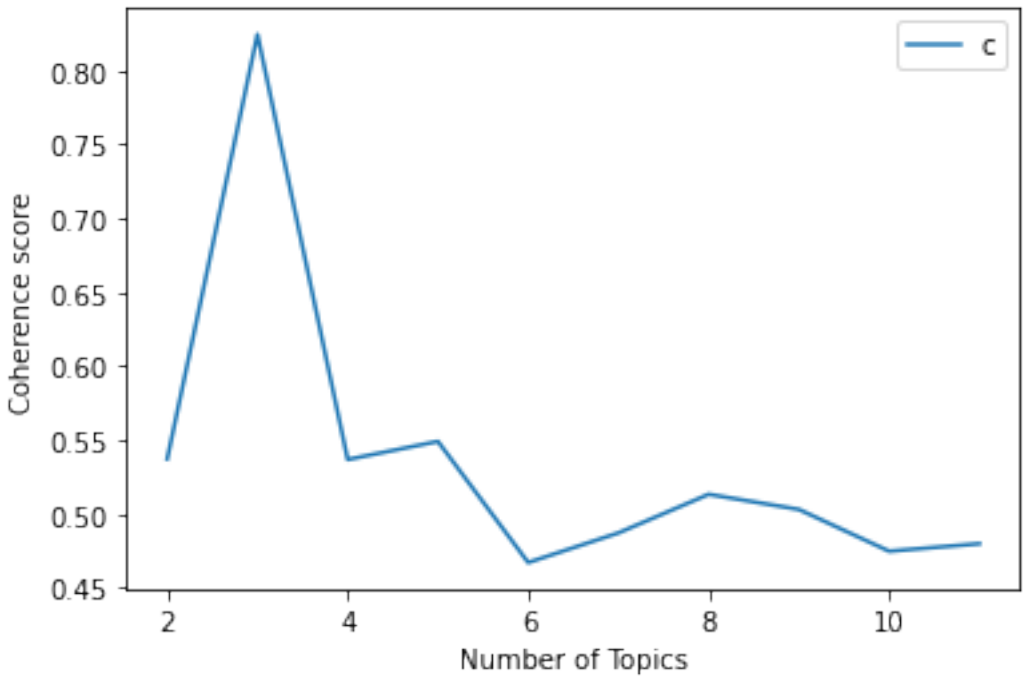

Figura 5: Coherencia de los modelos LSA en función del número de topics.

A primera vista podemos observar que muchos resultados son prácticamente idénticos. Aparentemente se podrían fusionar en un solo tema muchas de estas categorías. Sin embargo, lo más indicado será hacer varios modelos con distinto número de topics y quedarnos con el que tenga mayor coherencia.

El experimento se realizó con valores entre dos y doce y el resultado fue el de la figura 5. Apreciamos qeu el LSA de tres temas ostenta la máxima coherencia.

Si creamos el modelo óptimo, los temas resultantes son los siguientes.

- Tema 1: Elecciones presidenciales de EEUU de 2016 (trump, say, said, would, clinton, peopl, one, campaign, year, time, like, go, eu, vote, new...)
- Tema 2: Premier League (citi, v, h, 2016, 2017, unit, west, manchest, apr, dec, trump, leicest, liverpool, chelsea, palac...)
- Tema 3: Brexit (trump,clinton, eu, say, would, donald, leav, uk, republican, cameron, brexit, britain, govern, hillari, state...)

Sigue existiendo cierto solapamiento entre temas de artículos. Es evidente que el año de las elecciones de Estados Unidos la prensa anglosajona mencionaba a Trump hasta cuando se hablaba de la Premier League. Ahora bien, no cabe duda de que en este modelo las categorías se encuentran mejor definidas que antes.

8. Conclusiones

El fin último de este trabajo era presentar un modelo de análisis semántico, en concreto un clasificador de documentos por tema. Hemos visto que estos modelos tienen numerosas aplicaciones en el mundo real, como resumir curriculums, optimizar motores de búsqueda, sistemas de documentación, etc. Además hemos expuesto un modelo, cuya base matemática (SVD) la aprendimos en clase, y que se ha podido aplicar a un conjunto real de artículos con buenos resultados. La primera conclusión que podemos extraer de todo esto es que el análisis de componentes principales tiene más aplicaciones de las que podría parecer. En segundo lugar, nos gustaría recalcar la importancia de ajustar el modelo con el propósito de obtener los mejores resultados, ya que hemos pasado de tener una clasificación con siete categorías difusas a contar con tres bien definidas. En tercer lugar y bajo nuestro punto de vista la parte de documentación y aprendizaje fue la más costosa de este proyecto, una vez afianzados los conceptos teóricos, la implementación no fue demasiado compleja.

A la vista de algunas limitaciones que presenta el modelo LSA, el siguiente paso, de cara a futuros trabajos podría consistir en elaborar otros modelos como PLSA, LDA o lda2vec y comparar los resultados con los de LSA. A grandes rasgos, PLSA emplea la descomposición en valores singulares de la matriz de probabilidad, en vez de la matriz término-documento, donde cada celda contiene la co-ocurrencia del término y el documento [4]. LDA es la versión bayesiana de PLSA que sigue la Distribución de Dirichlet [1] y lda2vec es un modelo de deep learning basado en word2vec y LDA [5].

Referencias

[1] Andrew Y.; Jordan Michael I Blei, David M.; Ng. Latent dirichlet allocation. 2003.

[2] Datacamp. Conjunto de artículos. `https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Latent+Semantic+Analysis+in+Python/articles+4.txt` [Último acceso 6 de Noviembre de 2021].

[3] Yoav Goldberg and Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan Claypool Publishers, 2017.

[4] Thomas Hofmann. Probabilistic latent semantic analysis, 2013.

[5] Christopher E Moody. Mixing dirichlet topic models and word embeddings to make lda2vec, 2016.

[6] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, page 399–408, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Shaheen Syed and Marco Spruit. Full-text or abstract? examining topic coherence scores using latent dirichlet allocation. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 165–174, 2017.