Máster en Ciencia de Datos e Ingeniería de Computadores

UNIVERSIDAD DE GRANADA

Aprendizaje No Supervisado y Detección de Anomalías

— Trabajo Final: *Clustering* —

Analysis of TED Talks topics using document clustering

Santiago González Silot

Enero, 2023

Curso 2022-23

sgonzalezsilot@correo.ugr.es

Índice

1. Introducción

Para este proyecto final de *clustering* he decidido aplicar mis conocimientos previos de Procesamiento del Lenguaje Natural (NLP o PLN) junto a mis conocimientos recién adquiridos de *clustering*. Dentro del Procesamiento del Lenguaje Natural existen 3 tareas las cuales están especialmente relacionadas con el *clustering*.

1. En primer lugar el ***text clustering*** es como comunmente se conoce a a hacer *clustering* con las palabras, en esta tarea lo que se busca es una obtener una representación de un conjunto de palabras (por ejemplo todas las palabras del castellano) apta para su posterior uso en otros modelos de *machine learning*. Es un proceso totalmente no supervisado en el que finalmente se obtiene una representación vectorial de N dimensiones. Esta representación numérica que se obtiene como resultado del *clustering* se conoce en NLP como *Word embeddings* y se utilizan en distintas tareas de NLP debido a la dimensionalidad gigantesca que nos presenta un idioma y su imposibilidad para procesarlo utilizando técnicas clásicas como *One-Hot-Encoding* o varibles *Dummys*. Existen diversos *Word embeddings* conocidos en la literatura como pueden ser *word2vec* o *glove*.

2. En segundo lugar el ***document clustering*** es como el nombre indica el *clustering* de diversos documentos o textos utilizando el texto como "predictor". Para llevar a cabo esta tarea las palabras se han de preprocesar previamente utilizando un *word embedding* o utilizando la conocida técnica del *tf-idf* la cual indica como de relevante es una palabra para un documento en concreto dentro de un conjunto de documentos. Para ello se puden usar diversas técnicas clásicas de *clustering* como KMeans o DBSCAN, adaptando las medidas de distancia en caso de que sea necesario según la representación.

3. Finalmente y un poco más alejado del *clustering* esta el ***topic modelling*** el cual es una técnica que trata de encontrar temas dentro de un conjunto de documentos. Si bien podrían considerarse los temas de un conjunto de documentos los que resultasen de los *clusters* obtenidos en *document clustering* (y posteriormente algun análisis de relevancia de términos), en *topic modelling* esto no es así. En contraparte, el resultado de aplicar *topic modelling* no es un conjunto de *clusters* si no un conjunto de temas con las palabras más relevantes que describen a estos temas junto a un porcentaje de relevancia. La técnica más popular de topic modeling es LDA.

Durante este trabajo aplicará *document clustering* a un conjunto de 4005 charlas TED transcritas, con el objetivo de crear distintos *clusters* que agrupen las distintan temáticas de las que suelen tratar estas charlas, por lo que además también se podría considerar que se va a realizar en parte un leve descubrimiento de temas. En primer lugar se preprocesaran los datos utilizando *tf-idf* y probando con distintos algoritmos de *clustering* ya que es el enfoque más directo y posteriormente para trabajar también con *text clustering* se utilizará un *word embedding* y se volverá a probar los algoritmos para medir el impacto de una correcta representación del texto a la hora de realizar un buen *clustering*. Además durante el trabajo se irán eligiendo las medidas de distancia y las medidas de calidad más adecuadas.

2. Experimentos

Como experimentos para este trabajo se resolverá este problema de *clustering* con distintos algoritmos usando principalmente 2 enfoques. Un enfoque basado en tf-idf y otro enfoque basado en *embeddings*. Para cada uno de los enfoques se utilizarán los mismos algoritmos y para cada algoritmo se probarán con distintos parámetros con el objetivo de lograr el mejor resultado posible.

Además para la evaluación de estos *clusters* se analizarán paralelamente distintas medidas las cuales posteriormente se analizarán sus resultados y un razonamiento de el porqué de estos y la relación entre ellos.

2.1. Utilizando *tf-idf*

En primer lugar se ejecutarán los experimentos usando tf-idf, el Tf-idf (Term frequency - Inverse document frequency), es una medida numérica que expresa cuán relevante es una palabra para un documento en concreto dentro de un conjunto de documentos. Es una medida típica en el ámbito de la recuperación de información y minería de datos. Este valor aumenta proporcionalmente al número de veces que aparece una palabra en un documento pero se ve compensado por la frecuencia de esa palabra en el resto de documentos. Su formulación matemática es la siguiente:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Siendo $tf(t, d)$ la frecuencia del término t en un documento d:

$$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

Donde:

- $f_{t,d}$ es el número de veces que aparece t en un documento d.
- $\sum_{t' \in d} f_{t',d}$ es el número total de términos en el documento d.

Siendo $idf(t, D)$ la frecuencia de documento inversa, es decir, una medida de cuanta información proporciona una palabra, sobre como de común o específica es en todos los documentos.

$$idf(t, D) = \log \frac{N}{1 + |d \in D : t \in d|}$$

Donde:

- N es el número total de documentos en el corpus, $N = |D|$
- $|d \in D : t \in d|$ es el número de documento en los que el término t aparece.

Por tanto usaremos el valor del tf-idf de las palabras de los documentos como entrada para los algoritmos de *clustering*. Hay que tener en cuenta que cada palabra obtendrá un valor único por lo que la dimensionalidad del vector de entrada para los algoritmos será del número de palabras distintas del conjunto de documentos. En este caso es de 68303, lo cual es un número muy elevado y algunos de los algoritmos a utilizar se verán afectados en su rendimiento debido al problema de la dimensionalidad. Este problema surge ya que al aumentar el número de dimensiones el volumen del espacio aumenta exponencialmente y los datos se vuelven dispersos por lo que se vuelve un reto agruparlos y sacar relaciones entre ellos. Además los tiempos de cómputo serán considerablemente grandes. Por ello (dentro de tf-idf) se comparán los resultados utilizando el valor tf-idf de las 68303 palabras frente a utilizar las k más representativas

Teniendo esto en cuenta (y como se verá posteriomente), otros enfoques como *embeddings* a priori van a ser más rápidos y efectivos. Aun así tf-idf se puede mejorar aplicando alguna técnica de reducción de dimensionalidad.

Un punto clave a la hora de resolver un problema de *clustering* es la correcta elección de la distancia a usar por los algoritmos. Para ello se ha de tener conocimiento sobre la temática a resolver (¿que se está midiendo?) y ¿como se ha de medir? (como de parecido son 2 items entre sí) Por tanto la distancia a utilizar será la distancia coseno. Esta distancia se suele utilizar cuando se trabaja con *tf-idf* ya que no es sensible al tamaño de los textos. Si bien la típica distancia euclidea mide la distancia entre 2 puntos, la distancia coseno (o similaridad coseno) mide el angulo entre ellos. En este medida no es tan importante si en alguna dimensión el valor es 0 (si alguna palabra no aparece).

Si utilizasemos *td-idf* con la distancia euclidea es probable que un algoritmo agrupase juntos un texto de 100 palabras sobre ciencia con otro sobre tenis de 100 palabras, que con otro sobre ciencia pero de 500 palabras. Aun así la distancia coseno y euclidea están relacionadas siempre y cuando los vectores tengan una normalización 'l2'. Por lo que los de aquí en adelante para los algoritmos que no admitan modificar su distancia se utilizará la distancia euclidea pero con los vectores normalizados. La relación es la siguiente:

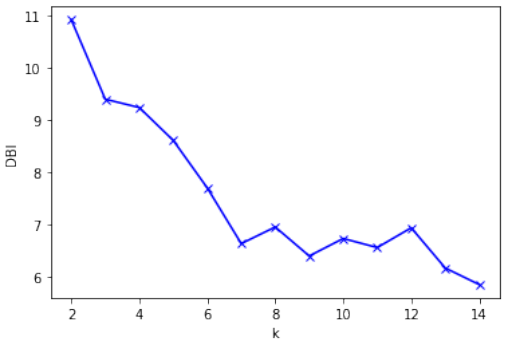
$$\text{Squared Euclidean distance} = 2 \cdot \text{Cosine Distance}$$

Por lo que para estos casos se elevará al cuadrado el vector denso y se dividirá por 2. Si bien el valor numérico exacto no será el mismo, se mantendrá el orden y la proporcionalidad, lo cual es lo que necesitamos de cara a los algoritmos.

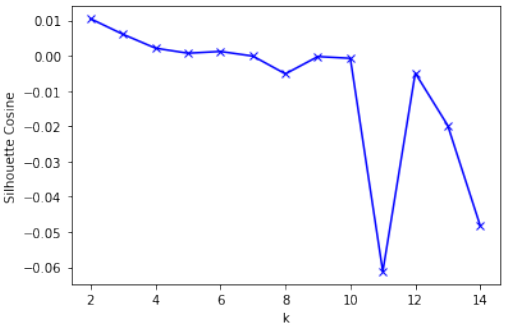
2.1.1. Kmeans

Para el Kmeans y el resto de algoritmos, se intentó utilizar la versión de sklearn por comodidad, pero se tuvo que utilizar otra versión distinta ya que la de sklearn no utiliza la GPU ni se puede paralelizar y utilizando la versión con un vector denso tan grande es necesario el uso KMeans que utilice GPU. Para ello se utilizó la versión de cuML que es una librería de NVIDIA dedicada a la implementación de algoritmos de machine learning en GPU.

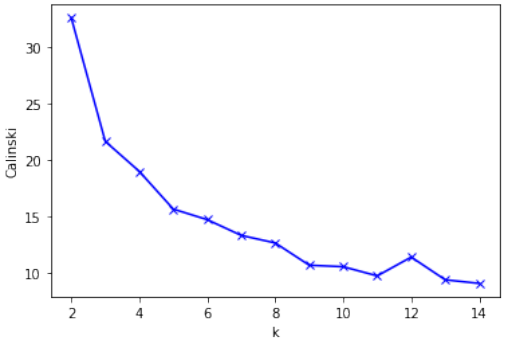
Para las pruebas se probó con disintos valores de k, entre 2 y 15 y se repitió cada ejecución (para no verse tan afectados con la aleatoriedad) 25 veces. Aun utilizando una versión que utilice una GPU y utilizando una GPU (proporcionada por la versión de Colab) Nvidia Tesla T4, los tiempos de ejecución fueron muy altos, en torno a 1 hora.



(a) DBI



(b) Silhoutte



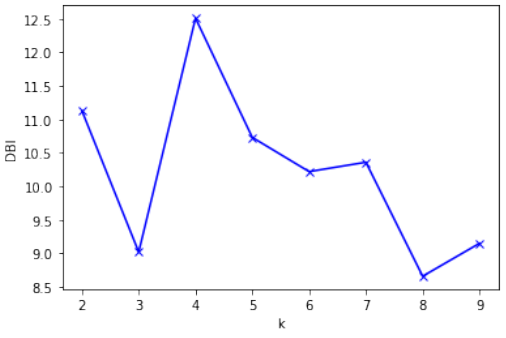
(c) Calinski

Figura 1: Evolución de los distintos K.

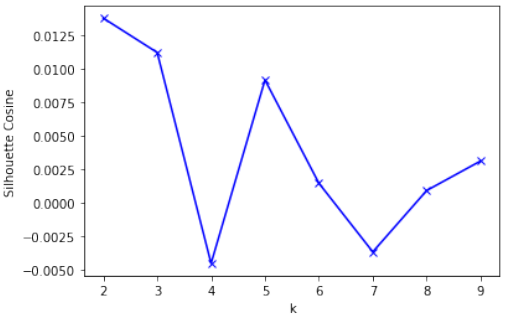
En este caso los resultados de DBI son casi estrictamente decreciente, por lo que por ese punto de vista sería buena idea intentar probar con un K mayor, aun así vemos una especie de codo entorno a $K = 7$ por lo que escogeremos este valor como el del mejor modelo. Además el valor de Silhoutte empeora considerablemente y al menos en $K = 7$ el valor de Silhoutte no es menor que 0. Además el valor de Calinski tampoco mejora conforme aumenta el K, si que empeora constantemente, prefiriendo agrupaciones más grandes.

2.1.2. Gaussian Mixture Models

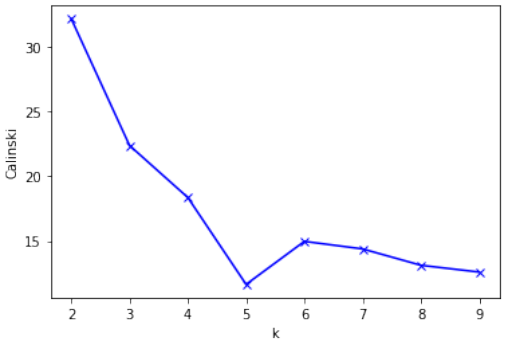
El segundo algoritmo a probar fue Gaussian Mixture Models ya que al tratarse de un modelo basado en mezcla de distribuciones Gaussianas permite solapamiento entre *clusters* lo cual puede llegar a ser muy interesante en este problema en concreto ya que al tratarse de texto sobre charlas de diversas temáticas, es normal que una charla pueda considerarse que trata más de una temática por tanto perteneciendo a varios *clusters*.



(a) DBI



(b) Silhoutte

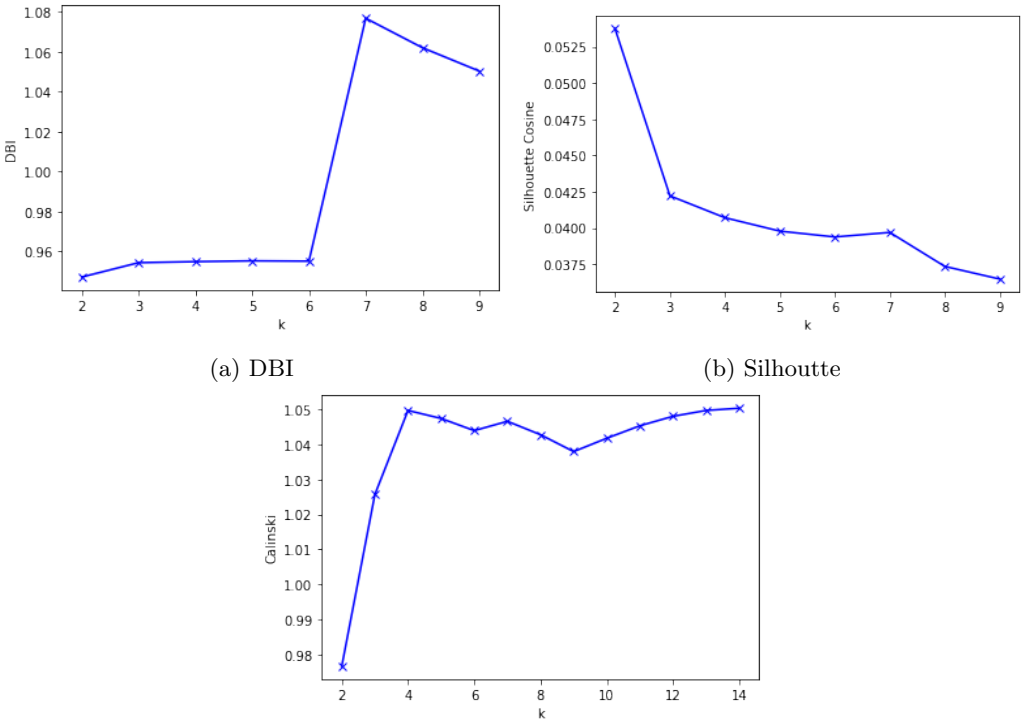


(c) Calinski

Figura 2: Evolución de los distintos K.

En este caso la linea de tendencia no es tan decreciente y tienen variación, el valor mejor teniendo en cuenta unicamente el DBI sería cuando $k = 8$ pero si tenemos en cuenta el valor de Silhouette y Caslinski vemos que quizá con 3 *clusters* es mejor.

2.1.3. Agglomerative *clustering*



(a) DBI

(b) Silhoutte

(c) Calinski

Figura 3: Evolución de los distintos K.

En este caso los valores parecen a primera vista muy buenos pero al ver que no varian casi da que pensar que algo raro debe de estar ocurriendo. Si nos fijamos cluster a cluster podemos ver que está todo el rato asignando casi todos los items a 1 único cluster, eso nos da muy buenos valores pero entendiendo el problema sabemos que esto no es una buena agrupación, por lo que no tendría mucho sentido usar este modelo.

2.1.4. Conclusiones parciales

Finalmente hemos visto que por un lado los valores de KMeans y GMM son relativamente parecidos, con la diferencia de que KMeans sigue una tendencia decreciente pero GMM tiene la ventaja de los conjuntos con solapamiento. Por otro lado Silhouette no nos aporta nada de información, aunque esto lo analizaremos en profundidad al final del estudio. Además Calisnki tampoco es de gran utilidad al ser casi siempre constantemente decreciente.

2.2. Utilizando *word embeddings*

Si bien con *tf-idf* hemos explicado anteriormente que la distancia a utilizar sería la coseno debido a que no se ve afectada por la longitud de los textos ni la frecuencia de las palabras, en este caso es más correcto utilizar la distancia euclidea ya que el vector que tenemos es simplemente los pesos de una red neuronal, y por tanto se ha de calcular la distancia entre 2 puntos y no una dirección.

Por otro lado el embedding a utilizar es sumamente importante, en este caso se utilizará el embedding de Bert, el cual es uno de los modelos de lenguaje más potente de los últimos años desarrollado por Google y que se ha demostrado ampliamente su potencia a la hora de resolver distintas tareas como *Masked Language*, *Sentence Classification* o *Sentence Similarity* (el cual nos interesa) entre otras tantas tareas.

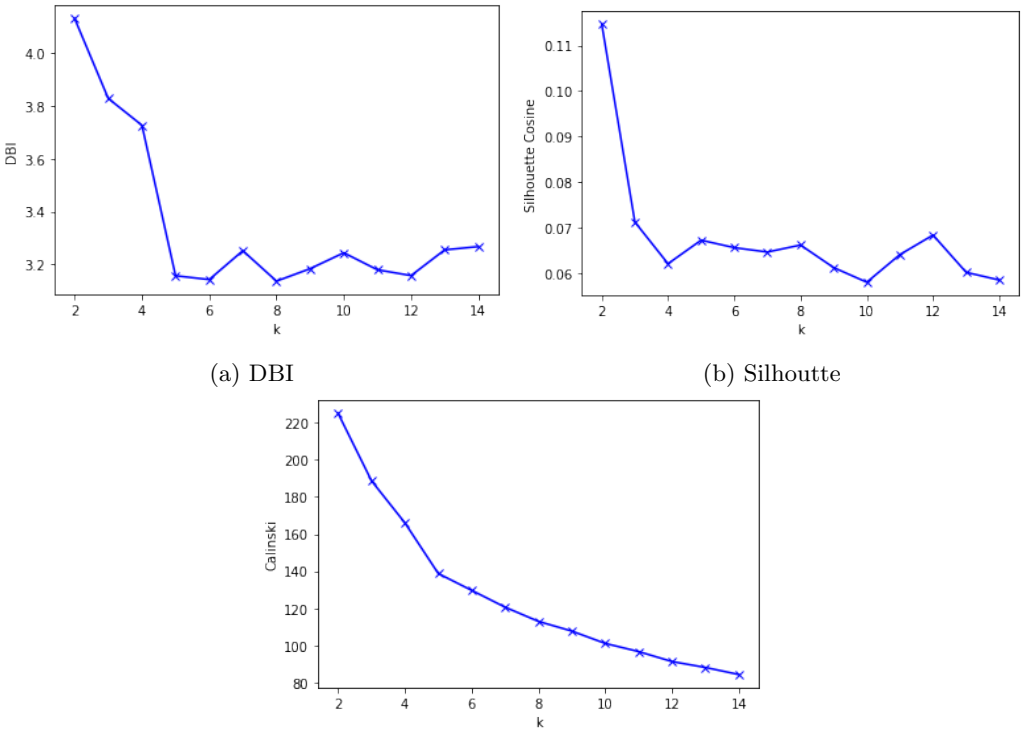
En este caso el tamaño del vector denso es considerablemente menor, si antes con *tf-idf* era de más de 68 mil características ahora es de al, lo cual supondrá unos tiempos de computo considerablemente menores junto a un esperable mejor rendimiento de los algoritmos por 2 razones.

1. Al tener muchas menos características se suaviza el problema de la dimensionalidad, lo cual hará que muchos algoritmos de *clustering* funcionen mejor y puedan agrupar mejor los documentos.

2. Estas características se podrían considerar como unas "super-features" vitaminadas las cual representan la información del lenguaje de mucha mejor forma que el *tf-idf* (ocurrencia de palabras) y de forma más compacta, aporta así más información con menos columnas.

A continuación se muestran los resultados de los algoritmos ejecutados nuevamente utilizando en este caso *embeddings* para ver si se cumplen las hipótesis.

2.2.1. KMeans



(a) DBI

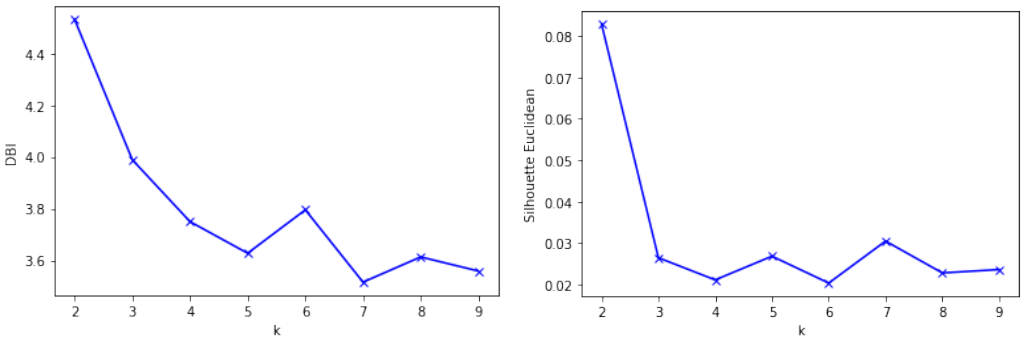
(b) Silhoutte

(c) Calinski

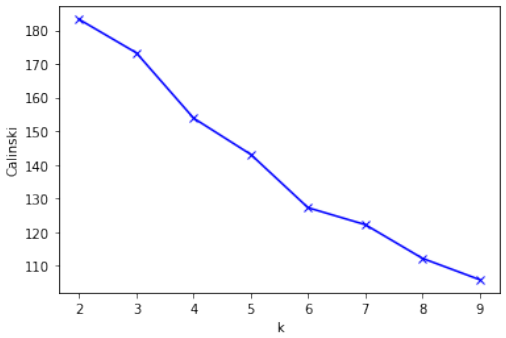
Figura 4: Evolución de los distintos K.

Como podíamos esperar KMeans ahora aporta mucho mejores valores de DBI que cuando utilizabamos *tfidf*, además podemos observar una tendencia decreciente clara y un codo bastante visible cuando hay 5 *clusters* con menos de 3,2 de DBI, por lo que a priori esta sería la mejor elección de k. Por otro lado el valor de Silhoutte ahora con *embeddings* es mejor pero sigue siendo muy bajo aunque algo más estable, lo cual es bueno. De nuevo Calinski vuelve a ser un valor constatemente decreciente pero considerablemente más alto que con *tf-idf*.

2.2.2. Gaussian Mixture Models



(a) DBI (b) Silhoutte

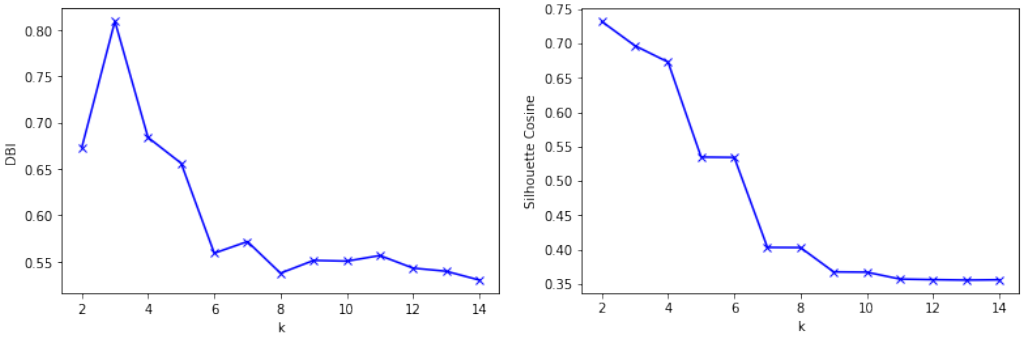


(c) Calinski

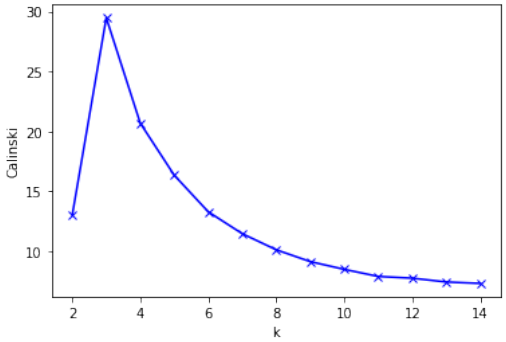
Figura 5: Evolución de los distintos K.

De nuevo los valores son mucho mejores que cuando se usaba *tf-idf*, en este caso la linea sí es decreciente claramente pero es más complicado observar un codo, podría ser cuando K es 5 o cuando es 7. Por otro lado aunque Silhoutte sigue sin ser alto, al menos ahora es estable. En este caso el valor de Calinski vuelve a ser decreciente pero no alcanza niveles tan bajos.

2.2.3. Agglomerative *clustering*



(a) DBI (b) Silhoutte



(c) Calinski

Figura 6: Evolución de los distintos K.

De nuevo Agglomerative *clustering* está introduciendo casi todos los items en 1 único cluster, esto nos da buenos valores, pero entendiendo el problema sabemos que no es una buena agrupación. Esto hace pensar que el problema no es por la codificación si no por el algoritmo en si y su funcionamiento con este tipo de datos de alta dimensionalidad.

3. Trabajo extra

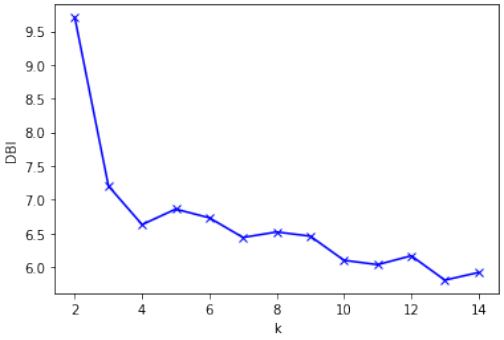
A modo de trabajo extra y con el fin de explorar aun más en las distintas opciones que nos aporta el *document clustering* se van a repetir los experimentos anteriormente mencionados con ciertas variaciones para *tf-idf* y *text embeddings*.

3.1. Trabajo extra 1: Seleccionar las palabras k más representativas del *tf-idf*.

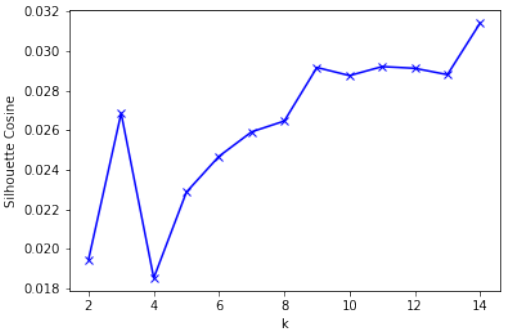
Como primer trabajo extra se van a repetir los experimentos pero en este caso utilizando solo el valor del *tf-idf* de las palabras que aparezcan en al menos un 5 % de los documentos y en menos del 50 % de los documentos. Con esto logramos 2 cosas, por un lado seleccionar las palabras más relevantes e intentar hacer agrupaciones más inteligentes a la hora de formar los *clusters* y por otro lado reducirconsiderablemente el tamaño del vector denso de 68303 a 1666 lo cual facilitará y mejorará el resultado de los algoritmos.

A continuación se muestran los resultados de las pruebas junto a unas pequeñas conclusiones.

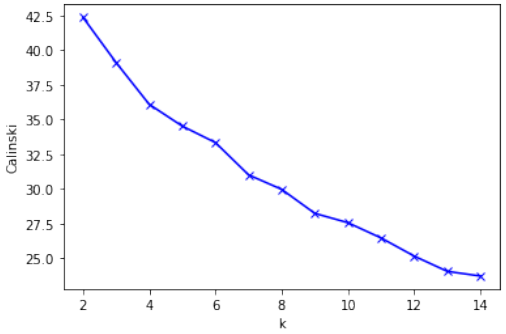
3.1.1. KMeans



(a) DBI



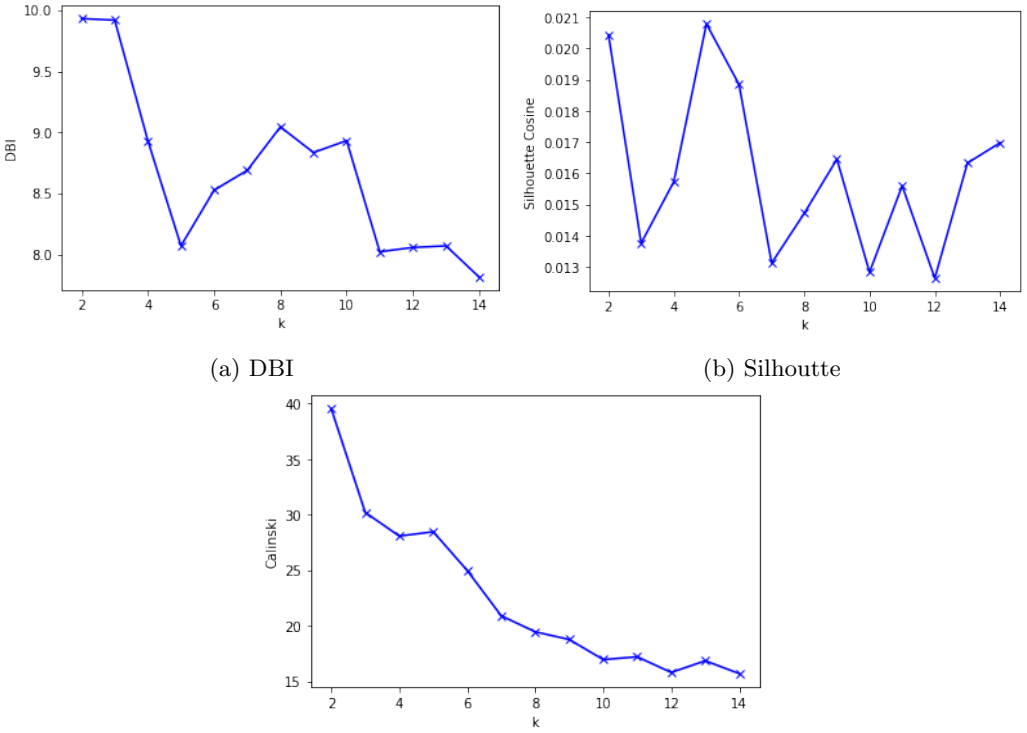
(b) Silhoutte



(c) Calinski

Figura 7: Evolución de los distintos K.

3.1.2. Gaussian Mixture Models



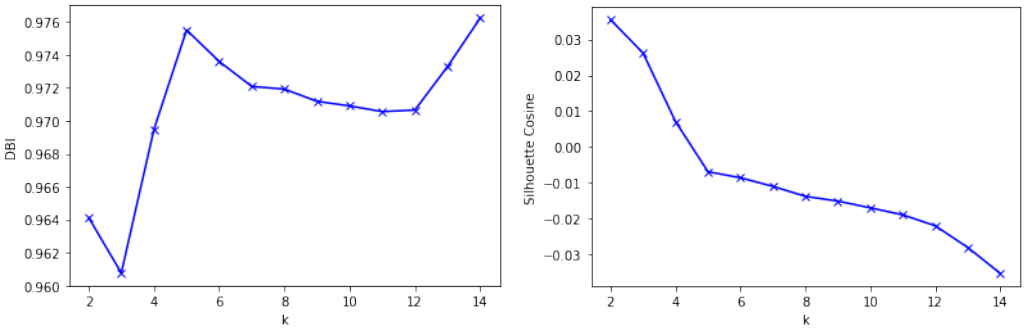
(a) DBI

(b) Silhoutte

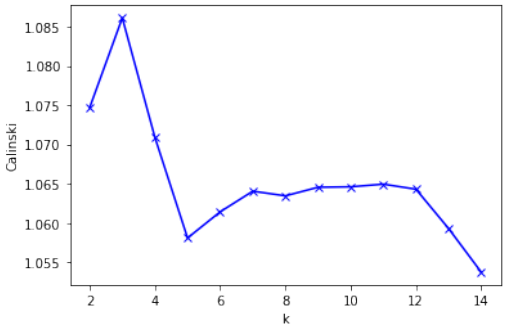
(c) Calinski

Figura 8: Evolución de los distintos K.

3.1.3. Agglomerative *clustering*



(a) DBI (b) Silhoutte



(c) Calinski

Figura 9: Evolución de los distintos K.

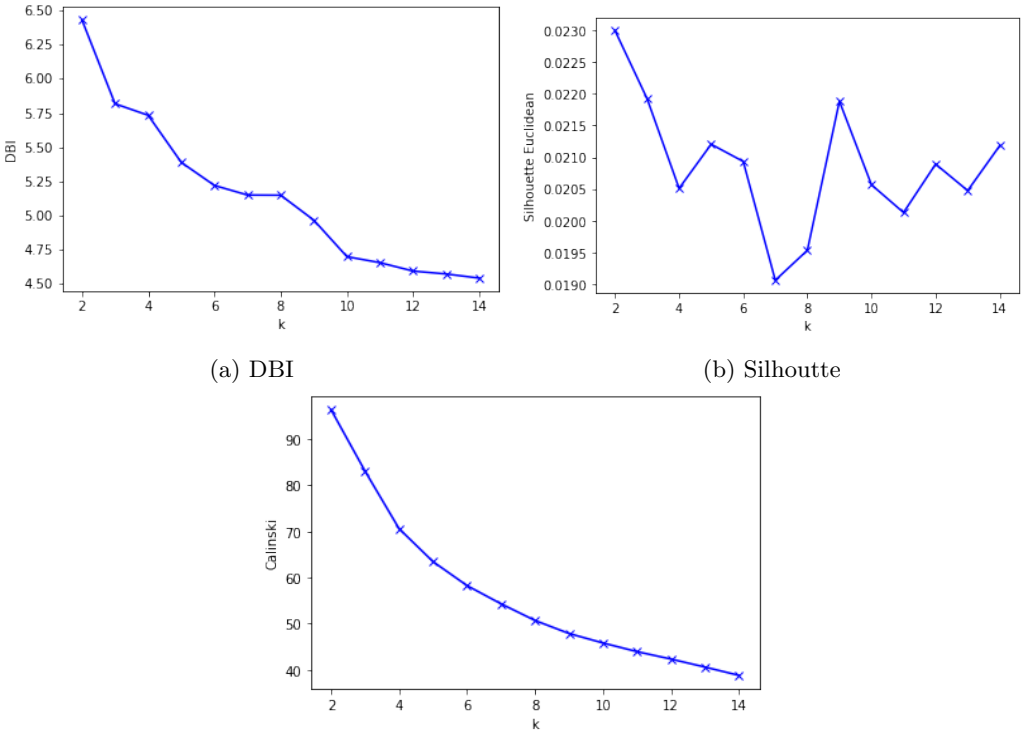
Como podemos observar las métricas para los distintos K mejoran levemente tanto de DBI como de Calisnki, además se puede apreciar de forma un poco más clara la tendencia decreciente, lo que puede dar a pensar que los *clusters* que son algo más estables. Con este experimento por un lado hemos visto una leve mejora con la versión original del *tf-idf* y por otro hemos visto una gran mejora en memoria y tiempos de computos durante la ejecución, siendo casi instantanea al tratarse de un vector denso mucho más pequeño (aunque siga siendo considerablemente grande).

3.2. Trabajo extra 2: Prueba con *text-embeddings* más potentes

Para este trabajo extra se repetiran de nuevo todos los experimentos utilizando el modelo de lenguaje MiniLM-L12-H384 de Microsoft. Este modelo de lenguaje según la librería SentenceTransformers aporta los mejores resultados y tiempos para tareas de *sentence similarity*, lo cual es clave para aplicar *document clustering*. En este caso tenemos un vector denso de 384, justo la mitad de pequeño que la versión base de BERT.

A continuación se muestran los resultados de las pruebas junto a unas pequeñas conclusiones.

3.2.1. KMeans



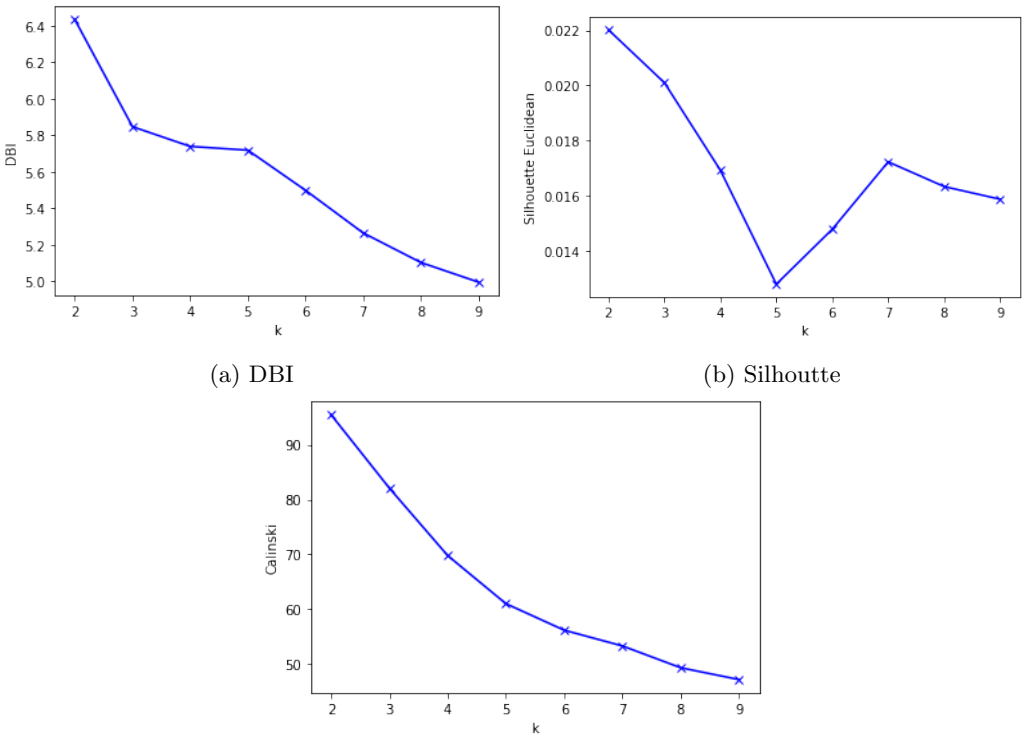
(a) DBI

(b) Silhoutte

(c) Calinski

Figura 10: Evolución de los distintos K.

3.2.2. Gaussian Mixture Models



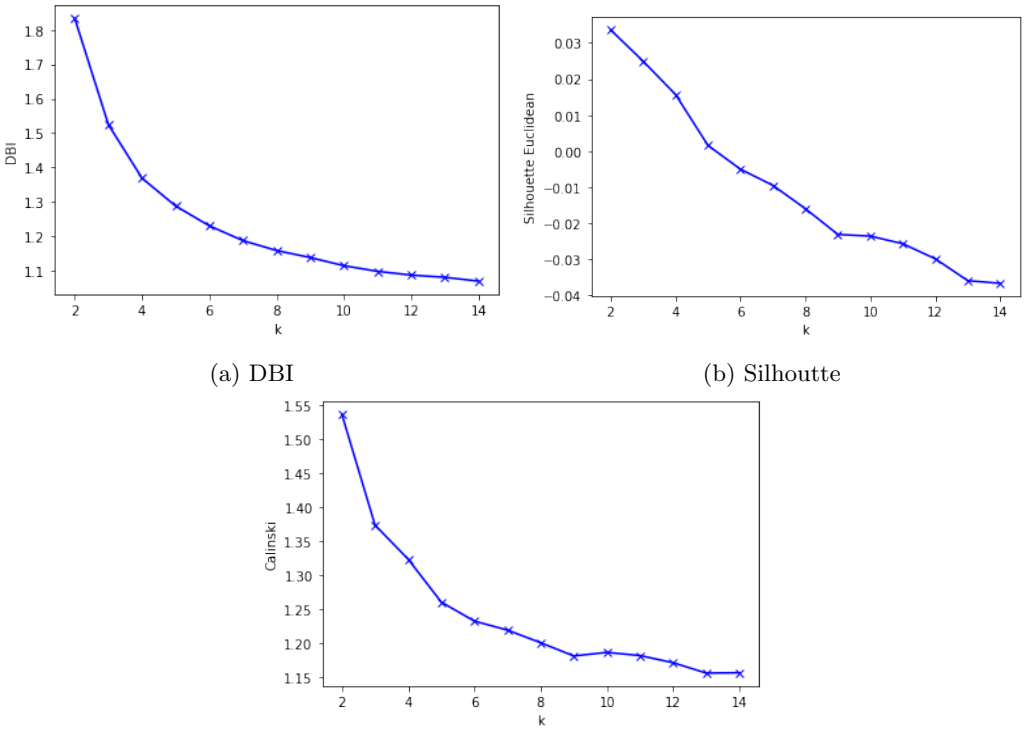
(a) DBI

(b) Silhoutte

(c) Calinski

Figura 11: Evolución de los distintos K.

3.2.3. Agglomerative *clustering*



(a) DBI

(b) Silhoutte

(c) Calinski

Figura 12: Evolución de los distintos K.

En este caso curiosamente no se mejoran los resultados si no que se empeoran levemente, aunque este modelo de lenguaje a priori sea más potente es probable que los resultados sean peores debido a que el vector denso es de la mitad de tamaño y no logre definir ni representar correctamente la información.

4. Visualizaciones

Con el fin de poder analizar de mejor forma los *clusters* y su calidad, vamos a proceder a pintar una nube de palabras y N-gramas para cada cluster que haya hecho cada algoritmo. De esta forma podremos ver las palabras y N-gramas más repetidos y representativos y así ver de que temática son las charlas del *cluster*. Para ello se utilizarán las mejores versiones obtenidas, es decir, las de KMeans y GMM usando *embeddings*.

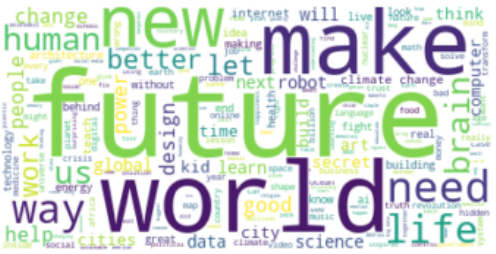
4.1. Visualizaciones de KMeans

Usaremos el modelo con K=5, es decir con 5 *clusters*. A continuación veamos en primer lugar las nubes de palabras para así tener una idea por encima sobre las temáticas de los diferentes *clusters*.

(a) Cluster 1

(b) Cluster 2

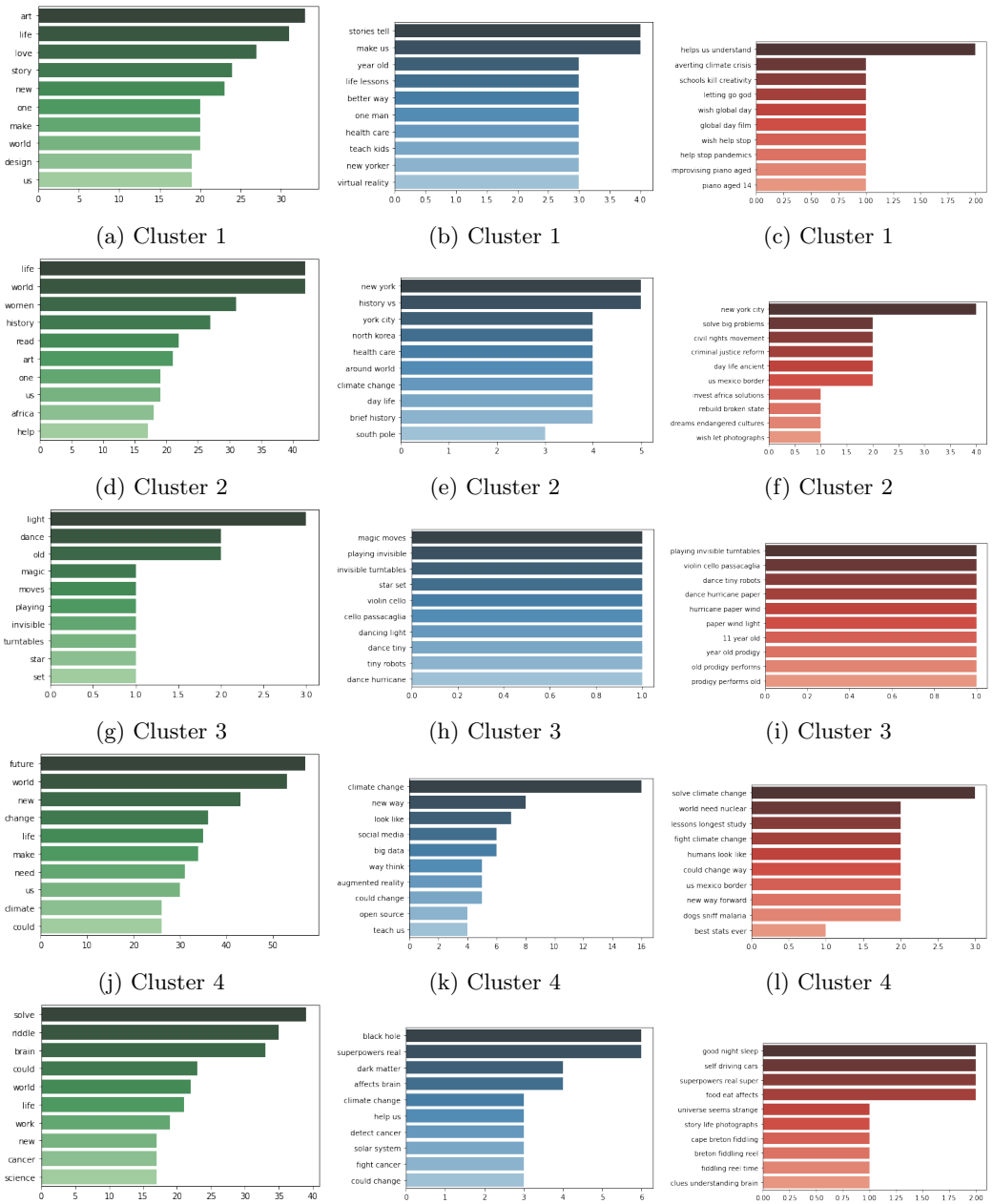
(c) Cluster 3



(d) Cluster 4

(e) Cluster 5

Figura 13: Nubes de palabras de los clusteres generador por GMM con *embeddings*.



(a) Cluster 1 (b) Cluster 1 (c) Cluster 1

(d) Cluster 2 (e) Cluster 2 (f) Cluster 2

(g) Cluster 3 (h) Cluster 3 (i) Cluster 3

(j) Cluster 4 (k) Cluster 4 (l) Cluster 4

(m) Cluster 5 (n) Cluster 5 (ñ) Cluster 5

Figura 14: Unigramas de los clusteres generador por GMM con *embeddings*.

Con todas estas gráficas podemos empezar a definir las temáticas de los *clusters*:

1. Este primer cluster parece que trata principalmente sobre arte, creatividad, diseño, etc. Además aparecen palabras como "lovez "story"que pueden indicar que este cluster no sea solo de arte si no de contar historias o hacer cosas que apasionan o ama la gente.

2. Este es el cluster más complicado de identificar como 1 única temática, por un lado destacan palabras como arte, vida y amor pero por otro bigramas como "stories tell" y "make us". Por tanto parece ser que este cluster trata sobre **historias personales de vida y de distintas temáticas variadas**.

3. En este cluster destacan palabras como "world" y "life", probablemente sean charlas sobre el mundo y la vida, quizá mas **filosóficas**. Además aparecen palabras como "africa" y "women"

por lo que quizá también haya charlas sobre **feminismo** o **pobreza en África**. En general parece que este cluster abarca la temática de los **derechos humanos** y **justicia**.

4. Este cluster parece tratar sobre **arte**, con bastante enfoque en **instrumentos músicales y géneros**. Es curioso que también destaque el trigrama "dance tiny robots" el cual podría dar la sensación que se han colado charlas sobre tecnología pero no, este trigrama aparece por una charla en concreto sobre coreografías musicales hechas por robots.

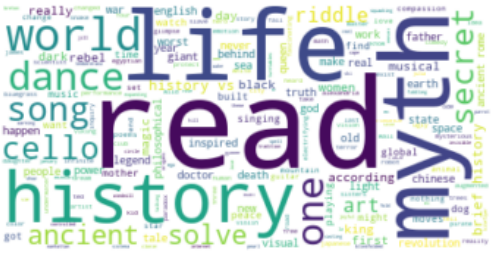
5. En este cluster de nuevo aparece la palabra "world" pero en este caso acompañada de "future", "solve climate change" y "world need nuclear", por lo que parece que tiene relación con la **ciencia, las necesidades humanas y el cambio climático**.

6. Este ultimo cluster parece tratar sofre **ciencia e inovación (I+D)**, aparecen terminos como "solve", "black hole" o "affects brain" entre otros.

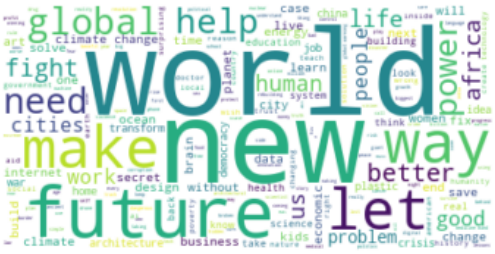
4.2. Visualizaciones de Gaussian Mixture Models

A continuación repetiremos el mismo proceso con los *clusters* generados por GMM, que, en este caso de nuevo son 5.

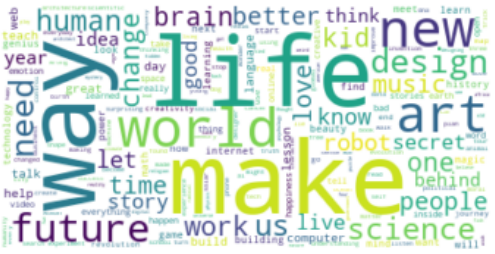
(a) Cluster 1



(b) Cluster 2



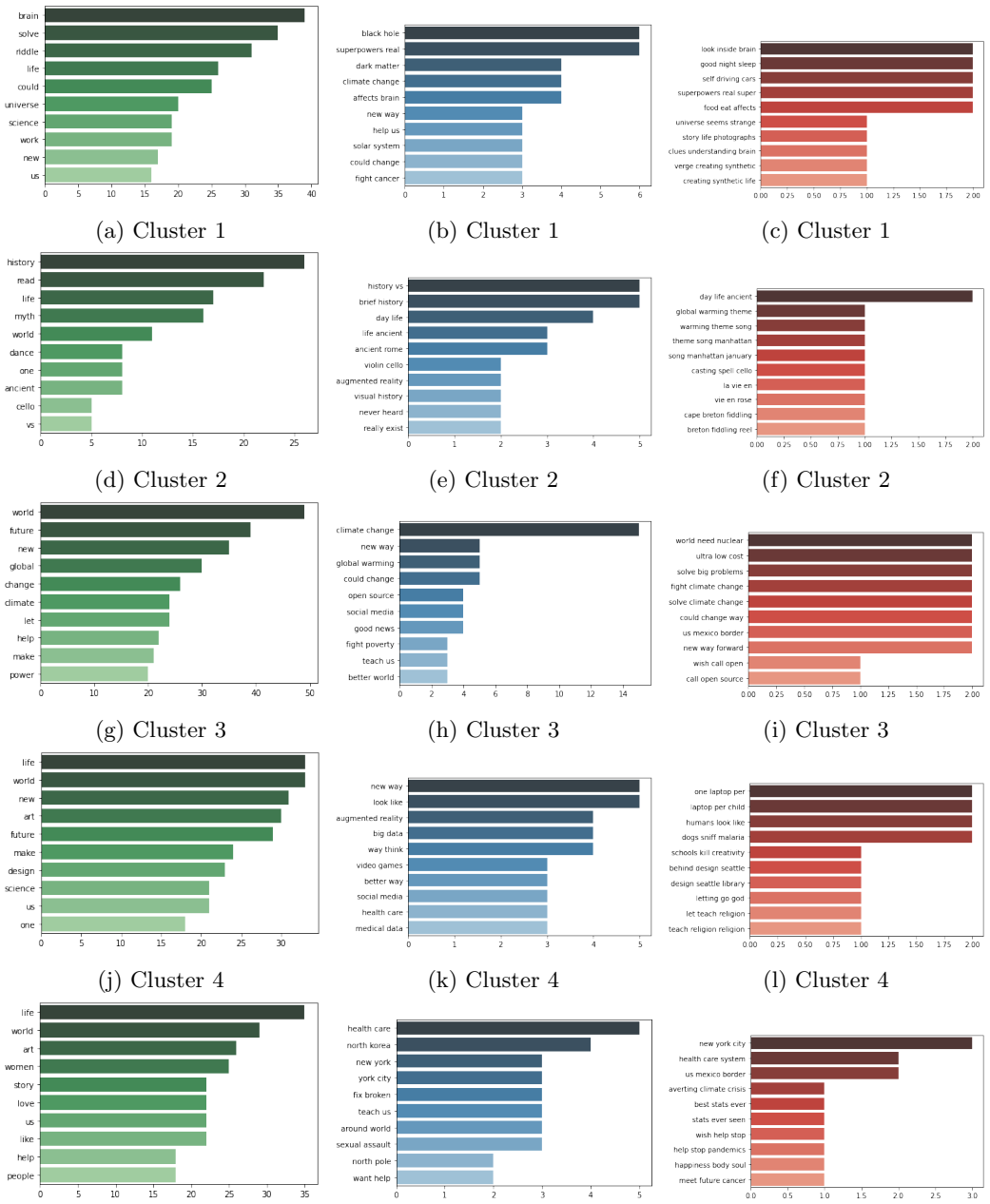
(c) Cluster 3



(d) Cluster 4

(e) Cluster 5

Figura 15: Nubes de palabras de los clusteres generador por GMM con *embeddings*.



(a) Cluster 1 (b) Cluster 1 (c) Cluster 1

(d) Cluster 2 (e) Cluster 2 (f) Cluster 2

(g) Cluster 3 (h) Cluster 3 (i) Cluster 3

(j) Cluster 4 (k) Cluster 4 (l) Cluster 4

(m) Cluster 5 (n) Cluster 5 (ñ) Cluster 5

Figura 16: Unigramas de los clusteres generador por GMM con *embeddings*.

Con todas estas gráficas podemos empezar a definir las temáticas de los *clusters*:

1. En este caso este primer cluster parece ser sobre **investigación y actualidad en la ciencia**, podemos verlo en términos como “look inside brain”, “ self driving cars” o “riddle” (enigma).
2. Este segundo cluster parece ser algo más disperso, se ve una parte claramente **musical**, pero por otra parte parece que hay partes de historia y actualidad.
3. Este cluster parece ser sobre el cambio climático, energía nuclear y geopolítica y el mundo en general. Esto se puede ver en términos como “world need nuclear”, “solve big problems” o “us mexico border” entre otros.
4. De nuevo este cluster parece estar relacionado con la **ciencia**, pero parece ser desde un punto de vista más **aplicado**. Esto se ve en “dogs sniff malaria” o “augmented reality”.

5. Finalmente este cluster parece bastante más variado, aparecen términos como "health care system", "sexual assault", "best stats ever" o "meet future cancer". Por tanto se podría categorizar este cluster como **sanidad y relacionados**.

5. Conclusiones y trabajo futuro

Finalmente en esta sección se hara un breve resumen de en que ha consistido el trabajo, las conclusiones que se han obtenido y posibles trabajos futuro.

En este trabajo se han utilizado conocimientos avanzados sobre NLP y *clustering* para así hacer agrupaciones de más de 4000 charlas TED. Se ha trabajado con las 3 formas prototípicas en las que se ven el NLP y el *clustering* unidos. Por un lado se han hecho document *clustering* de las transcripciones de las charlas y así poder agruparlas por similaridad, también se ha utilizado text *clustering* al utilizar una representación del texto basado en *embeddings* y finalmente se ha hecho en cierta forma topic modeling ya que se han analizado los posibles temas al analizar las palabras más frecuentes de los *clusters* obtenidos.

Para hacer el document *clustering* se han seguido 2 enfoques distintos y los más usados en este tipo de tareas como son el *tf-idf* y los *embeddings*, además se han probado distintos algoritmos como son KMeans, Gaussian Mixture Models y Agglomerative *clustering*.

Finalmente para cada uno de estos modelos se han escogido los mejores parámetros según el análisis de métodos intrínsecos como lo son el DBI, Silhouette y el Calinski and Harabasz Score.

Además como trabajo extra se han repetido los experimentos utilizando un subconjunto de los términos más relevantes de *tf-idf* y se ha probado con un modelo de lenguaje más potente para los *embeddings*.

5.1. Conclusiones

Durante el desarrollo de este trabajo se han llegado a varias conclusiones y puntos clave, los cuales son mencionados a continuación:

- Se ha visto la relevancia que tiene el problema de la dimensionalidad en problemas de NLP al tratarse de datos con una alta dimensionalidad. Se ha visto que los modelos sufren (en tiempo de cómputo y resultados) a la hora de trabajar frente a este tipo de datos. Además se ha comprobado que una representación más pequeña (y más efectiva) de los textos ayuda en gran medida a que se ejecuten más rápido los algoritmos y puedan agrupar más fácilmente. Esto se ha visto a la hora de trabajar con vectores densos de gran tamaño, en los cuales "todo está alejado de todo" ya que crece exponencialmente las distancias conforme crecen las dimensiones y dificulta el correcto funcionamiento de los algoritmos.

- Uno de los puntos clave a la hora de enfrentarse a un problema de *clustering* es la correcta elección de la medida de distancia, en este caso se ha visto y se ha razonado en que momento y porqué (entendiendo los datos) es mejor la distancia euclidea y cuando es mejor la distancia coseno. Esto es un punto clave ya que para obtener un resultado exitoso y que los algoritmos funcionen se tiene que escoger correctamente la medida teniendo en cuenta los datos.

- Como ya se ha mencionado antes, la representación con *embeddings* ha funcionado considerablemente mejor que la que utiliza *tf-idf*, esto no solo es por el problema de la dimensionalidad si no por que los *embeddings* son una mejor representación de los textos que el *tf-idf* la cual aporta información de mayor calidad y esto ayuda a los algoritmos de *clustering* a poder encontrar dichas agrupaciones más facilmente.

- Por otro lado se ha visto la importancia del uso de versiones más avanzadas de los algoritmos los cuales utilicen paralelización por GPU lo cual se vuelve clave cuando se trabaja con vectores tan densos. Para ello se han utilizado las librerías orientadas al uso de técnicas de *machine learning* con GPU como lo son *cuML* y *PyCave*.

- Se ha visto que la metríca Silhoutte no ha aportado un gran valor en este trabajo (siempre se encontraba casi en 0.0) y eso es debido a que este métrica puntua como 1 cuando están

perfectamente definidos los *clusters*, como -1 cuando lo están mal (muy pocos o demasiados) y como 0 cuando está indefinido y hay mucho solapamiento entre *clusters*. Como ya se ha mencionado antes es normal que trabajando con este tipo de datos haya solapamiento entre *clusters* ya que puede que una charla trate sobre varios temas a la vez. Por otro lado hemos vistos que GMM considera que hay solapamiento y esto va de la mano con este tipo de datos y proporciona buenos resultados.

- Por otro lado el valor Calinski no mejora ya que al calcularse como el cociente de la suma de la dispersión entre *clusters* y de la dispersión dentro de *clusters*, conforme se aumenta el K probablemente este valor baja ya que en este problema los *clusters* se entremezclan.
- Si bien a priori Agglomerative *clustering* parecía el mejor algoritmo al aportar los mejores resultados, se ha contabilizado el número de items de cada cluster por separado y se ha visto que la distribución utilizada no tenía sentido para este problema ya que ponía casi todos los items en único cluster.
- Finalmente se ha visto la utilidad de técnicas como la nubes de palabras o los N-gramas para la valoración subjetiva de los *clusters* formados y obteniendo así una leve idea sobre las posibles temáticas de las agrupaciones. Utilizando estas técincas de cierta forma se está actuando con un cierto *ground truth* y evaluando la calidad de los *clusters*.

5.2. Trabajo futuro

Para finalizar este trabajo a continuación se presentan una serie de puntos clave que podrían servir como trabajo futuro y continuación del trabajo.

- Comparación con más métricas. Si bien las métricas que se han utilizado son de las métricas más comunes para *clustering*, se podría estudiar el uso de más métricas a la hora de repetir este estudio en el futuro.
- Por otro lado se podría probar con el uso de más modelos de *clustering* como pueden ser DBSCAN o HDBSCAN.
- También podrían compararse más representaciones como lo pueden ser el uso de *embeddings* de otro modelo de lenguaje como puede ser GPT-2 en vez de Bert, o una compración de distintos modelos de lenguaje para "sentence similarity". También podría hacerse un ajuste más fino con distintos hiperparámetros a la hora de usar el *tf-idf* probando con usar un vector denso aun más pequeño o una técnica más avanzada de selección de términos clave.