Clustering de noticias provenientes de distintas fuentes

Mallku E. Sodevila Raffa

5 de marzo de 2017

Clustering de noticias provenientes de distintas fuentes

- Descripción del problema.
- Formulación de la solución.
- Implementación.
- Resultados.
- Trabajos a futuro.

Descripción del problema

- Problema a resolver[1]:

[1] Basado en "*Incremental Clustering of News Reports*", Joel Azzopardi and Christopher Staff, www.mdpi.com/journal/algorithms

Descripción del problema

- Problema a resolver[1]:
  - Se obtienen artículos de distintos portales digitales de noticias, de manera continua.

[1] Basado en "*Incremental Clustering of News Reports*", Joel Azzopardi and Christopher Staff, www.mdpi.com/journal/algorithms

Descripción del problema

- Problema a resolver[1]:
  - Se obtienen artículos de distintos portales digitales de noticias, de manera continua.
  - En la medida en la que llegan noticias, estas deben agruparse en *clusters*, de acuerdo al tema específico del cual tratan ("*fine-grained clustering*").

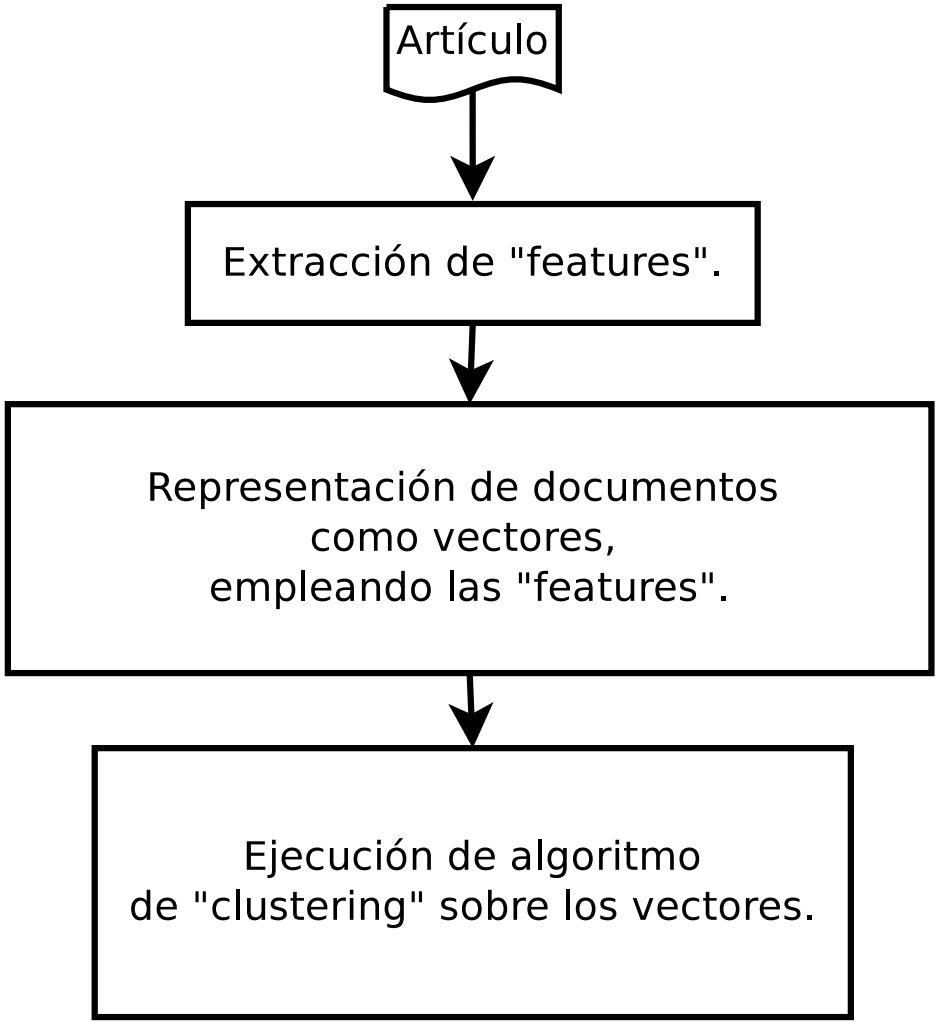
[1] Basado en "*Incremental Clustering of News Reports*", Joel Azzopardi and Christopher Staff, www.mdpi.com/journal/algorithms

Descripción del problema

- Problema a resolver[1]:
  - Se obtienen artículos de distintos portales digitales de noticias, de manera continua.
  - En la medida en la que llegan noticias, estas deben agruparse en *clusters*, de acuerdo al tema específico del cual tratan ("*fine-grained clustering*").
  - Por lo tanto, los clusters no se conocen de antemano. Se crean o crecen, en la medida en la que llegan nuevas noticias.

[1] Basado en "*Incremental Clustering of News Reports*", Joel Azzopardi and Christopher Staff, www.mdpi.com/journal/algorithms

Formulación de la solución



Formulación de la solución

- **Extracción de features**:
  - Subconjunto de palabras del documento, semánticamente significativas (removemos *stop-words*).

Formulación de la solución

- **Extracción de features**:
  - Subconjunto de palabras del documento, semánticamente significativas (removemos *stop-words*).
  - Extraemos la raíz de cada palabra.

Formulación de la solución

- **Extracción de features**:
  - Subconjunto de palabras del documento, semánticamente significativas (removemos *stop-words*).
  - Extraemos la raíz de cada palabra.
  - No consideramos el orden en el que ocurren las palabras (modelo de *Bag-of-words*).

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - **term frequency**: $\log\left(\frac{raw\_term\_freq_i}{doc\_size_j} + 1\right)$

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - *term frequency*: $\log\left(\frac{raw\_term\_freq_i}{doc\_size_j} + 1\right)$
  - **Intuición detrás de la fórmula**:
    - La ocurrencia de un término posiblemente ayude a determinar el contenido de un documento.

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - *term frequency*: $\log\left(\frac{raw\_term\_freq_i}{doc\_size_j} + 1\right)$
  - **Intuición detrás de la fórmula**:
    - La ocurrencia de un término posiblemente ayude a determinar el contenido de un documento.
    - Documento extenso, con poca proporción de ocurrencias de un término dado, debería dif. de un documento corto con gran proporción de ocurrencias del término $\Rightarrow$ normalizamos.

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - *term frequency*: $\log\left(\frac{raw\_term\_freq_i}{doc\_size_j} + 1\right)$
  - **Intuición detrás de la fórmula**:
    - La ocurrencia de un término posiblemente ayude a determinar el contenido de un documento.
    - Documento extenso, con poca proporción de ocurrencias de un término dado, debería dif. de un documento corto con gran proporción de ocurrencias del término $\Rightarrow$ normalizamos.
    - No está claro que sea preferible una relación lineal entre ocurrencia de un término e importancia para determinar el contenido del documento $\Rightarrow$ vamos por algo más suave que una relación lineal: log (la base del logaritmo no importa).

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - **inverse document frequency**: $\log\left(\frac{coll\_size}{doc\_freq_i + 1}\right)$

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - *inverse document frequency*: $\log\left(\frac{coll\_size}{doc\_freq_i + 1}\right)$
  - **Intuición detrás de la fórmula**:
    - Términos pocos frecuentes son "más informativos"sobre el contenido de un documento, que aquellos con gran frecuencia de ocurrencia.

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término i, dentro de un documento j, donde:
  - *inverse document frequency*: $\log\left(\frac{coll\_size}{doc\_freq_i + 1}\right)$
  - **Intuición detrás de la fórmula**:
    - Términos pocos frecuentes son "más informativos"sobre el contenido de un documento, que aquellos con gran frecuencia de ocurrencia.
    - Para morigerar el peso de este factor, tomamos el logaritmo.

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término *i*, dentro de un documento *j*, donde:
  - $TF.IDF = term\ frequency * inverse\ document\ frequency$

Formulación de la solución

- **Mapeo de documentos a vectores en el hiper-espacio**: calculamos *TF.IDF* (*term frequency - inverse document frequency*), para cada término *i*, dentro de un documento *j*, donde:
  - $TF.IDF = term\ frequency * inverse\ document\ frequency$
  - En la definición de cada factor, sumamos 1, así el estadístico queda definido para todo término, inclusive para aquellos que no ocurran en el documento.

Formulación de la solución

- Características de los vectores obtenidos:

Formulación de la solución

- Características de los vectores obtenidos:
  - **Dimensión:** cantidad de términos diferentes extraídas del total de documentos analizados.

Formulación de la solución

- Características de los vectores obtenidos:
  - Dimensión: cantidad de términos diferentes extraídas del total de documentos analizados.
  - **Los vectores son dispersos**: un documento contendrá sólo un subconjunto de los términos del total que observamos (con la llegada de nuevos artículos, quizás llegan nuevos términos).

Formulación de la solución

- Características de los vectores obtenidos:
  - Dimensión: cantidad de términos diferentes extraídas del total de documentos analizados.
  - Los vectores son *dispersos*: un documento contendrá sólo un subconjunto de los términos del total que observamos (con la llegada de nuevos artículos, quizás llegan nuevos términos).
  - **Orientación de los vectores**: componentes distintas de 0 en dimensiones referidas a palabras que aparezcan en los documentos que están representando. La orientación de los vectores conjuga inf. sobre los términos que el documento contiene y no contiene.

Formulación de la solución

- Características de los vectores obtenidos:
  - Dimensión: cantidad de términos diferentes extraídas del total de documentos analizados.
  - Los vectores son *dispersos*: un documento contendrá sólo un subconjunto de los términos del total que observamos (con la llegada de nuevos artículos, quizás llegan nuevos términos).
  - Orientación de los vectores: componentes distintas de 0 en dimensiones referidas a palabras que aparezcan en los documentos que están representando. La orientación de los vectores conjuga inf. sobre los términos que el documento contiene y no contiene.
  - **Magnitud de los vectores**: documentos con términos diferentes, pero semejantes en importancia y frecuencia, estarán representados por vectores de magnitud semejante.

Formulación de la solución

- Medidas de *similaridad* a usar:
  - **Distancia euclídea**: depende fuertemente de la magnitud de los vectores y no tanto de su orientación $\Rightarrow$ no tiene en cuenta, de manera adecuada, el contenido de los documentos.

Formulación de la solución

- Medidas de *similaridad* a usar:
  - Distancia euclídea: depende fuertemente de la magnitud de los vectores y no tanto de su orientación $\Rightarrow$ no tiene en cuenta, de manera adecuada, el contenido de los documentos.
  - **Cosine similarity**: coseno entre vectores. Captura mejor la semejanza de vectores en términos de la orientación de los mismos.

Formulación de la solución

- **Algoritmo de clustering**: variante del algoritmo *k-means*, adaptado a las necesidades de la tarea.

Formulación de la solución

- **Algoritmo de clustering**: variante del algoritmo *k-means*, adaptado a las necesidades de la tarea.
  - No se conocen, de antemano, la cantidad de *clusters* a generar.

Formulación de la solución

- **Algoritmo de clustering**: variante del algoritmo *k-means*, adaptado a las necesidades de la tarea.
  - No se conocen, de antemano, la cantidad de *clusters* a generar.
  - Cada nuevo artículo es comparado con los clusters existentes. Cada clúster es representado por el vector *centroide*: vector cuya componente *i-ésima* es la media aritmética de las componentes *i-ésimas* de los vectores dentro del cluster.

Formulación de la solución

- **Algoritmo de clustering**: variante del algoritmo *k-means*, adaptado a las necesidades de la tarea.
  - No se conocen, de antemano, la cantidad de *clusters* a generar.
  - Cada nuevo artículo es comparado con los clusters existentes. Cada clúster es representado por el vector *centroide*: vector cuya componente *i-ésima* es la media aritmética de las componentes *i-ésimas* de los vectores dentro del cluster.
  - Se calcula *cosine similarity* entre el vector que representa al documento y el centroide de cada clúster. Si es $\geq$ umbral predefinido $\Rightarrow$ se agrega el vector al clúster. Caso contrario: nuevo clúster, con el documento como único miembro.

Formulación de la solución

- **Algoritmo de clustering**: variante del algoritmo *k-means*, adaptado a las necesidades de la tarea.
  - No se conocen, de antemano, la cantidad de *clusters* a generar.
  - Cada nuevo artículo es comparado con los clusters existentes. Cada clúster es representado por el vector *centroide*: vector cuya componente *i-ésima* es la media aritmética de las componentes *i-ésimas* de los vectores dentro del cluster.
  - Se calcula *cosine similarity* entre el vector que representa al documento y el centroide de cada clúster. Si es $\geq$ umbral predefinido $\Rightarrow$ se agrega el vector al clúster. Caso contrario: nuevo clúster, con el documento como único miembro.
  - No hay reorganización de clusters.

Implementación

- **Implementación en python3**: cálculo del estadístico TF.IDF, algoritmo de clustering, suite de test y evaluación del rendimiento del algoritmo.

Implementación

- Implementación en python3: cálculo del estadístico TF.IDF, algoritmo de clustering, suite de test y evaluación del rendimiento del algoritmo.
- **Librerías extras**:
  - *nltk* (RegexpTokenizer, SnowballStemmer).

Implementación

- Implementación en python3: cálculo del estadístico TF.IDF, algoritmo de clustering, suite de test y evaluación del rendimiento del algoritmo.
- **Librerías extras**:
  - *nltk* (RegexpTokenizer, SnowballStemmer).
  - *scipy* (dok_matrix, para representación de vectores dispersos)

Implementación

- Implementación en python3: cálculo del estadístico TF.IDF, algoritmo de clustering, suite de test y evaluación del rendimiento del algoritmo.
- **Librerías extras**:
  - *nltk* (RegexpTokenizer, SnowballStemmer).
  - *scipy* (dok_matrix, para representación de vectores dispersos)
  - *scrapy* (para la obtención de un corpus de noticias previamente agrupadas, Google News, contra el cual medir el rendimiento del algoritmo; también para la implementación del programa final, que agrupa noticias extraídas de distintos portales.)

Resultados

- Se emplearon las medidas de *precission*, *recall* y *F1* para estimar la calidad de cada clúster construido.

Resultados

- Se emplearon las medidas de *precission*, *recall* y *F1* para estimar la calidad de cada clúster construido.
- Para cada clúster de referencia (del corpus de Google News), se buscó el clúster más próximo devuelto por la implementación evaluada.

Resultados

- Se emplearon las medidas de *precission*, *recall* y *F1* para estimar la calidad de cada clúster construido.
- Para cada clúster de referencia (del corpus de Google News), se buscó el clúster más próximo devuelto por la implementación evaluada.
- Se experimentó con distintos valores para el umbral de *semejanza* entre noticia y clúster.

Resultados

- Se emplearon las medidas de *precission*, *recall* y *F1* para estimar la calidad de cada clúster construido.
- Para cada clúster de referencia (del corpus de Google News), se buscó el clúster más próximo devuelto por la implementación evaluada.
- Se experimentó con distintos valores para el umbral de *semejanza* entre noticia y clúster.
- **Sin embargo...:** la evaluación de la calidad de los clusters obtenidos requirió también *inspección manual* (entonces, se trabajó con corpus relativamente pequeños: $\approx$ 200 noticias).

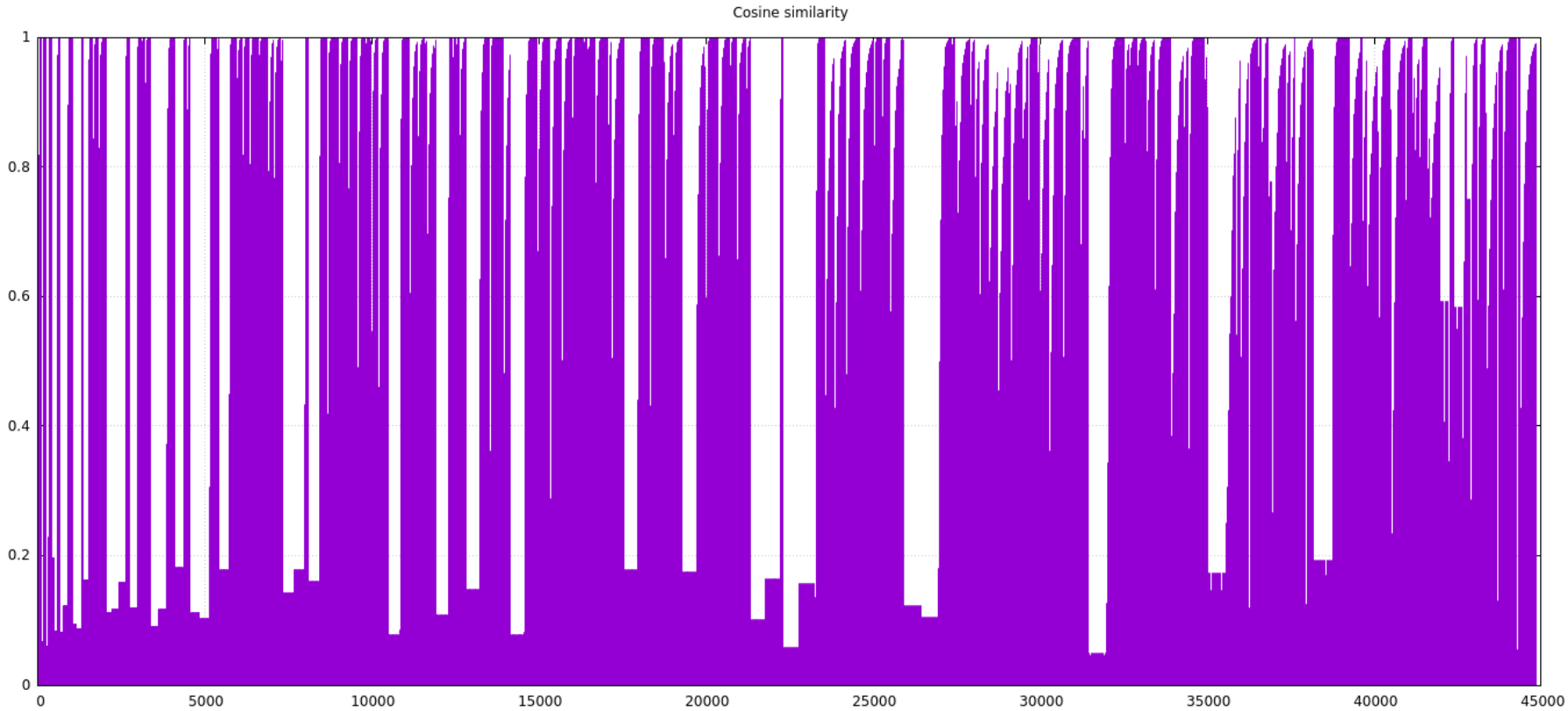
Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

Threshold	*Precision*	*Recall*	*F1*
0.2	0.679	0.718	0.698
0.25	0.679	0.718	0.698
0.3	0.679	0.718	0.698
0.35	0.679	0.718	0.698
0.4	0.679	0.718	0.698
0.45	0.679	0.718	0.698
0.5	0.679	0.718	0.698
0.55	0.679	0.718	0.698
0.6	0.679	0.718	0.698
0.65	0.679	0.718	0.698
0.7	0.680	0.718	0.698
0.75	0.680	0.718	0.698

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:



Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - Agrupa la noticia "*El film Aquarius cristaliza el desconcierto de un Brasil sin Dilma*"[2] junto con noticias sobre la destitución de Cunha[3] (tengo el RSS que demuestra esto...).

[2] lanacion.com.ar/1937071-el-film-aquarius-cristaliza-el-desconcierto-de-un-brasil-sin-dilma

[3] lavoz.com.ar/mundo/destituyeron-cunha-el-inquisidor-de-dilma-rousseff, www.clarin.com/mundo/Destituyen-Eduardo-Cunha-diputado-Dilma_0_1649835015.html

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - Agrupa la noticia "*El film Aquarius cristaliza el desconcierto de un Brasil sin Dilma*"[2] junto con noticias sobre la destitución de Cunha[3] (tengo el RSS que demuestra esto...).
  - Relaciona a la noticia "*Isabel Macedo y Mariano Martínez, juntos en una nueva novela*"[4](noticia 1) con "*El cruce entre Brancatelli y Urtubey en Intratables*"[5] (noticia 2). La palabra *Macedo* no aparecen en este último artículo.

[2] lanacion.com.ar/1937071-el-film-aquarius-cristaliza-el-desconcierto-de-un-brasil-sin-dilma

[3] lavoz.com.ar/mundo/destituyeron-cunha-el-inquisidor-de-dilma-rousseff, www.clarin.com/mundo/Destituyen-Eduardo-Cunha-diputado-Dilma_0_1649835015.html

[4] infobae.com/teleshow/infoshow/2016/09/13/isabel-macedo-y-mariano-martinez-juntos-en-una-nueva-novel

[5] clarin.com/politica/Brancatelli-cruzo-Urtubey-Intratables_0_rkFsdyS2.html

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - *cosine similarity* entre los artículos: 0.02367720842496972 1

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - *cosine similarity* entre los artículos: 0.02367720842496972 1
  - *cosine similarity* entre el centroide y noticia 1: 0.3258665232272106.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - *cosine similarity* entre los artículos: 0.02367720842496972 1
  - *cosine similarity* entre el centroide y noticia 1: 0.3258665232272106.
  - *cosine similarity* entre el centroide y la noticia 2: 0.8268899015910558.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - *cosine similarity* entre los artículos: 0.02367720842496972 1
  - *cosine similarity* entre el centroide y noticia 1: 0.325866523227210 6.
  - *cosine similarity* entre el centroide y la noticia 2: 0.826889901591055 58.
  - ¿La definición de *centroide* no es del todo adecuada para identificar una noticia en particular?: a medida que agregamos artículos a un clúster, agregamos términos nuevos al cómputo del centroide $\Rightarrow$ el mismo puede alejarse de los términos centrales que caracterizan a una noticia.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con los clusters de Google**:
  - Ejemplo: en el clúster sobre la pelea de Brancatelli y Urtubey, se incluye también una noticia[6] que sí contiene la palabra *Macedo*.

[6] infobae.com/teleshow/paso-en-la-tv/2016/09/13/diego-brancatelli-y-juan-manuel-urtubey-polemizaron-a

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con mis clusters**:

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con mis clusters**:
  - Semejante a la situación con los clusters de Google: clusters con noticias que no están directamente relacionadas entre sí, sino por gracia de una cadena de artículos que van enriqueciendo el centroide con nuevos términos claves. Aunque en mi resultados, hay mayor presencia de clusters grandes que abarcan noticias con pobre semejanza.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **Problemas con mis clusters**:
  - Semejante a la situación con los clusters de Google: clusters con noticias que no están directamente relacionadas entre sí, sino por gracia de una cadena de artículos que van enriqueciendo el centroide con nuevos términos claves. Aunque en mi resultados, hay mayor presencia de clusters grandes que abarcan noticias con pobre semejanza.
  - Obtengo 2 maneras diferentes de agrupar noticias (Google News y mis resultados), ambas con errores diferentes, pero de la misma naturaleza.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **¿Posible solución?**:

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **¿Posible solución?**:
  - Redefinir la noción de centroide: tomar solamente el promedio de los pesos de los términos que están presentes en todos los artículos del clúster.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

- **¿Posible solución?**:
  - Redefinir la noción de centroide: tomar solamente el promedio de los pesos de los términos que están presentes en todos los artículos del clúster.
  - En la medida en la que el clúster se agranda, más resaltamos el peso de los términos principales.

Resultados

Corpus con 219 noticias, agrupadas previamente por Google News:

Threshold	*Precision*	*Recall*	*F1*
0.2	0.781	0.783	0.782
0.25	0.784	0.783	0.784
0.3	0.801	0.794	0.797
0.35	0.852	0.818	0.835
0.4	0.866	0.800	0.832
0.45	0.866	0.814	0.849
0.5	0.895	0.786	0.837
0.55	0.921	0.741	0.821
0.6	0.924	0.670	0.780
0.65	0.938	0.644	0.764
0.7	0.936	0.594	0.727
0.75	0.974	0.483	0.646

Resultados

Corpus con 544 noticias, agrupadas previamente por Google News:

Threshold	*Precision*	*Recall*	*F1*
0.2	0.681	0.730	0.705
0.25	0.691	0.761	0.725
0.3	0.726	0.746	0.736
0.35	0.727	0.761	0.744
0.4	0.748	0.766	0.756
0.45	0.765	0.767	0.766
0.5	0.805	0.760	0.782
0.55	0.833	0.765	0.793
0.6	0.865	0.710	0.780
0.65	0.899	0.688	0.779
0.7	0.913	0.636	0.750
0.75	0.928	0.584	0.717

Trabajos a futuro

- Terminar de implementar una araña que descargue noticias de distintos portales y las agrupe según nuestro algoritmo (actualmente, por incompatibilidades entre Scrapy, python3 y pickle no está terminado).

Trabajos a futuro

- Terminar de implementar una araña que descargue noticias de distintos portales y las agrupe según nuestro algoritmo (actualmente, por incompatibilidades entre Scrapy, python3 y pickle no está terminado).
- ¿Paralelizar el clustering de noticias, estableciendo un criterio para determinar de antemano cuando 2 noticias no puede formar parte del mismo clúster?

Trabajos a futuro

- Terminar de implementar una araña que descargue noticias de distintos portales y las agrupe según nuestro algoritmo (actualmente, por incompatibilidades entre Scrapy, python3 y pickle no está terminado).
- ¿Paralelizar el clustering de noticias, estableciendo un criterio para determinar de antemano cuando 2 noticias no puede formar parte del mismo clúster?
- Terminar de implementar "*freezing*" de clústers viejos, para los cuales se asume que ya no van a entrar artículos nuevos.

Trabajos a futuro

- Terminar de implementar una araña que descargue noticias de distintos portales y las agrupe según nuestro algoritmo (actualmente, por incompatibilidades entre Scrapy, python3 y pickle no está terminado).
- ¿Paralelizar el clustering de noticias, estableciendo un criterio para determinar de antemano cuando 2 noticias no puede formar parte del mismo clúster?
- Terminar de implementar "*freezing*" de clústers viejos, para los cuales se asume que ya no van a entrar artículos nuevos.
- Alguna forma gráfica, fácil de interpretar, para visualizar el resultado y contrastar portadas de diarios digitales.