بازیابی پیشرفته اطلاعات

دکتر بیگی

گزارش فاز سوم پروژه

فاطمه هادی‌زاده ۹۵۱۰۵۹۰۲
فائزه پویامهر ۹۵۱۰۵۴۴۳
فاطمه باقری ۹۵۱۰۵۴۱۹

بخش ۱. خوشه‌بندی

پیش‌پردازش

در این بخش با استفاده از کتابخانه hazm روی بخش title و summary هر مستند پیش‌پردازش‌های لازم (شامل:نرمال کردن ، stem کردن و حذف لغات تکراری) انجام می‌شود و در نهایت خروجی آن یه لیستی از دیکشنری‌ها (هر دیکشنری معادل یه مستند است) با کلید های link،text و Tag می‌باشد.

Tf_idf و Word2vec:

برای تبدیل متون به فضای برداری از از این دو روش استفاده شده است .

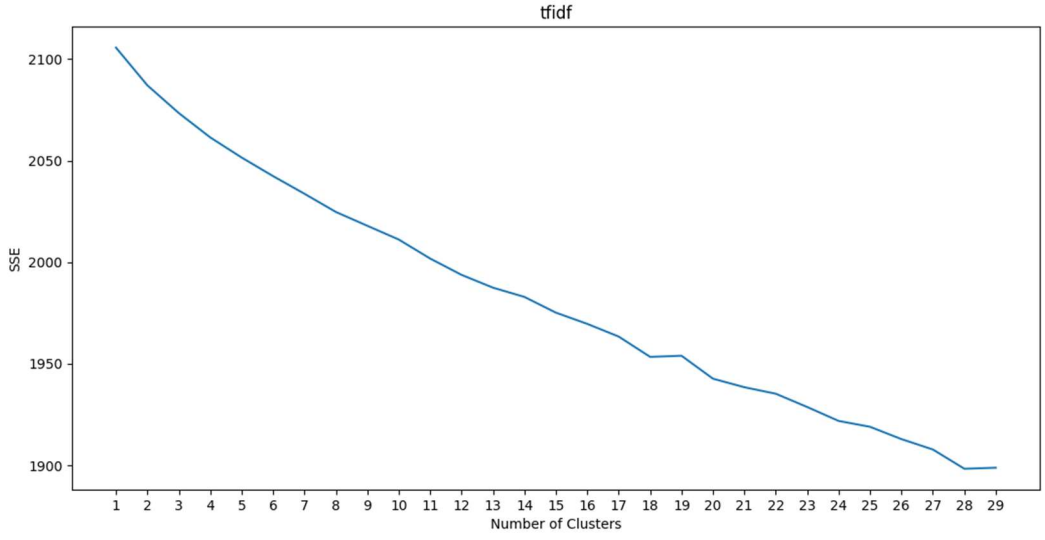
برای Tf_idf از کلاس Tf_idf_vectorizer موجود در فاز قبل استفاده شده که با ورودی گرفتن کل مجموعه مستندات یک لیستی از بردارهای tf_idf متناظر با هر مستند را خروجی می‌دهد.

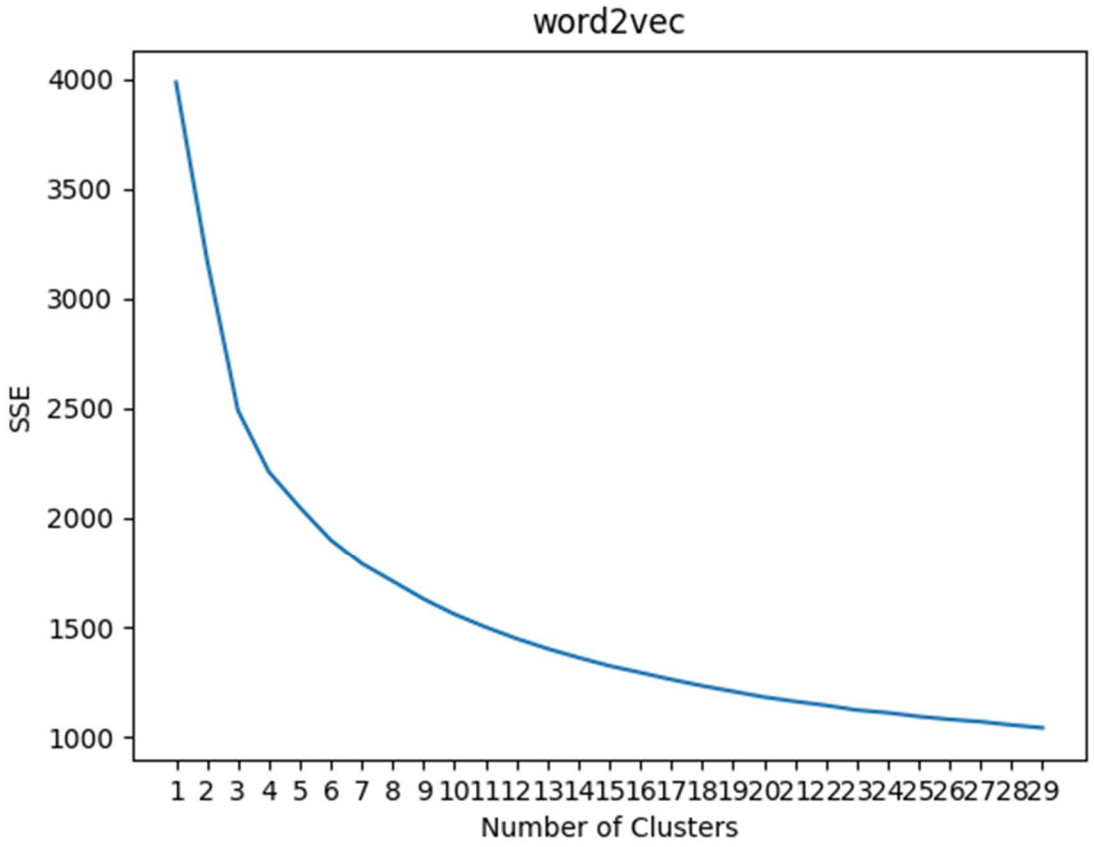
برای Word2vec هم از کتابخانه genism استفاده شده که با ورودی گرفتن بخش text مربوط به همه مستندات خروجی مشابه tf_idf برمی‌گرداند.

انتخاب پارامتر k:

برای انتخاب تعداد دسته، الگوریتم K-means را به ازای k های مختلف اجرا کرده و Sum Square Error را به ازای هر k محاسبه کرده و نمودار SSE را بر حسب k رسم می‌کنیم. با افزایش مقدار k مقدار SSE به طور پیوسته کاهش می‌یابد. زیرا با افزودن میانگین‌های بیشتر، فاصله بین هر نقطه با نزدیک‌ترین مرکز به آن کاهش می‌یابد. اما در نقطه‌ای از نمودار منحنی SSE شروع به خم شدن می‌کند که به عنوان نقطه elbow شناخته می‌شود . تصور می‌شود که مقدار x این نقطه یک trade-off منطقی بین خطا و تعداد خوشه‌ها باشد.

Output:






```
the elbow point for tfidf is: 18
the elbow point for word2vec is: 10
```

به طور میانگین تعداد خوشه را ۱۴ در نظر می‌گیریم. البته با شماردن تعداد تگ‌های اصلی داده‌ها می‌توان دید که ۱۴ تا تگ متمایز وجود دارد.

K-means .1.1

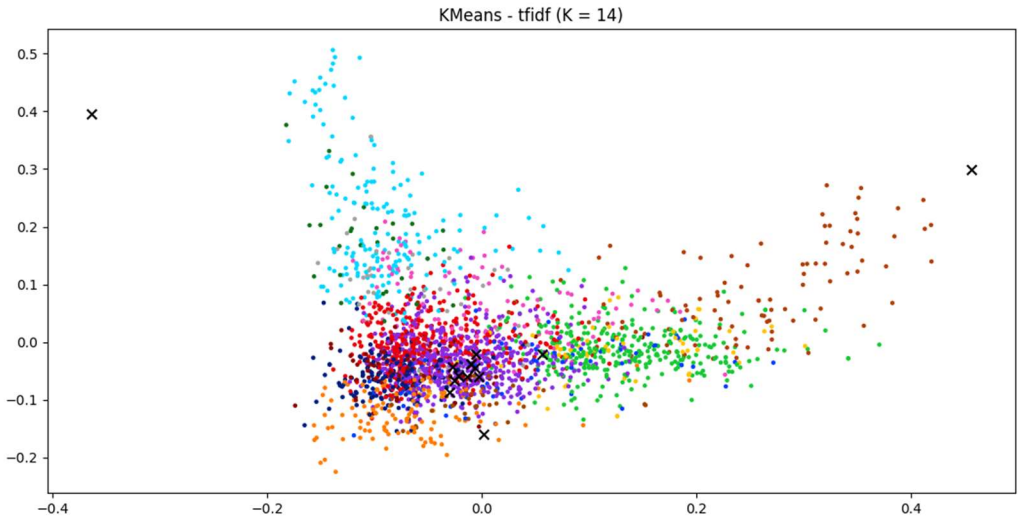
این الگوریتم تلاش می‌کند تا داده‌ها را به گونه‌ای خوشه‌بندی کند که مجموع فاصله نقاط هر خوشه تا میانگین خوشه کمینه شود و در هر مرحله از اجرا دو گام Assignment داده‌ها به خوشه‌ای که داده کمترین فاصله را تا میانگین آن دارد و Re-compute میانگین خوشه‌ها محاسبه می‌شوند. در ابتدا میانگین k خوشه به طور تصادفی یا با الگوریتم `k-means++` انتخاب می‌شود. برای پیاده‌سازی این الگوریتم از تابع آماده `sklearn.cluster.KMeans` در حالت `k-means++` استفاده شده است. در زیر نتیجه خوشه‌بندی با استفاده از این الگوریتم به تفکیک روش‌های تبدیل به فضای برداری tf-idf, Word2vec آمده است:

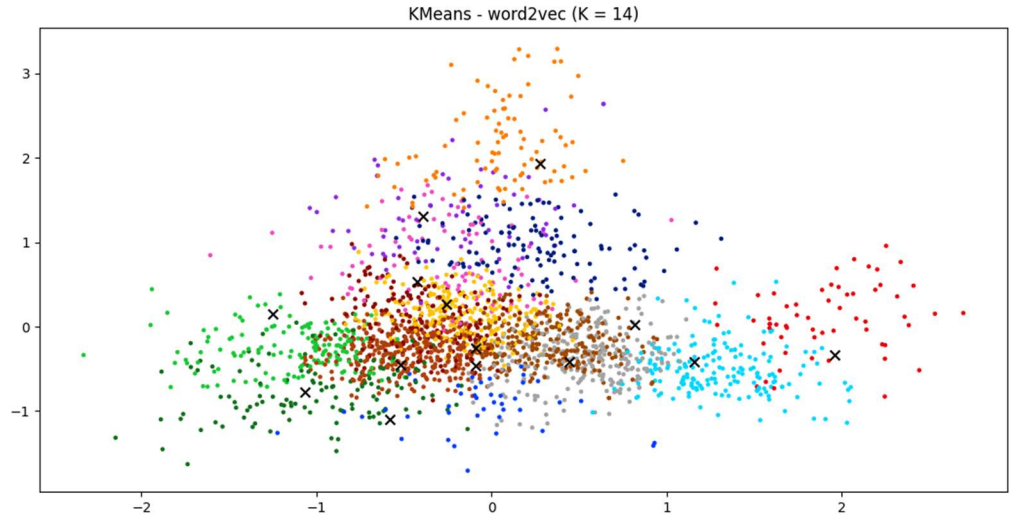
Output:

```
-> KMeans clustering algorithm and tfidf vectorizer
   Purity score: 0.5056740807989106 , Rand Index score: 0.8063329544428517 ,
Adjusted Mutual Info score: 0.29566723531919753 , Adjusted Rand Index score:
0.13853593053198462
-> KMeans clustering algorithm and word2vec vectorizer
   Purity score: 0.4316840671811167 , Rand Index score: 0.8166945165600702 ,
Adjusted Mutual Info score: 0.20960416994043002 , Adjusted Rand Index score:
0.11207738552055697
```

معیارهای Purity, RI, ARI و AMI محاسبه شده است. دو روش تبدیل تقریبا مانند هم عمل می‌کنند. tfidf کمی بهتر عمل کرده است.

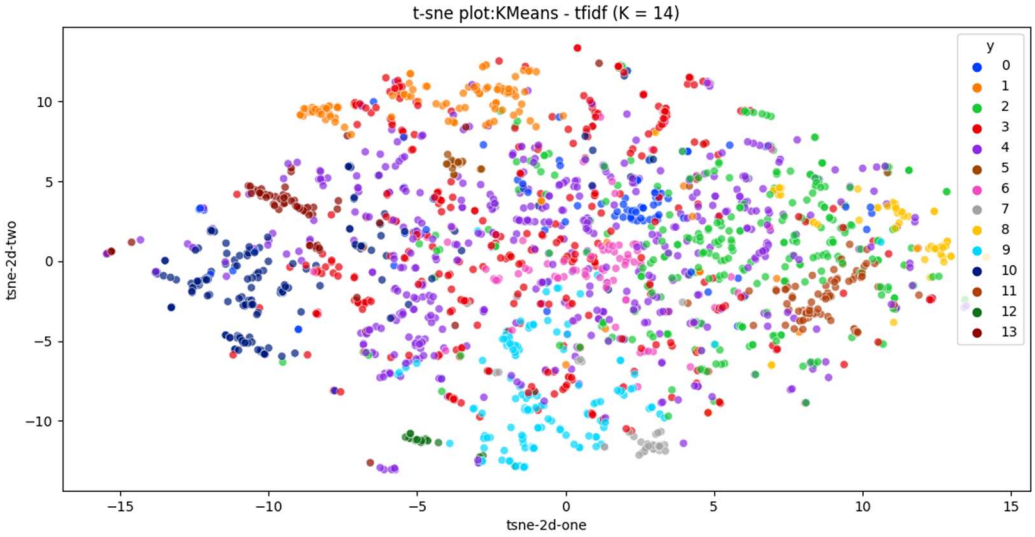
در شکل‌های زیر ابتدا داده‌ها را با استفاده از الگوریتم PCA کاهش بعد داده سپس آن‌ها را در نمودار دو بعدی plot کردیم. هر رنگ نشان‌دهنده‌ی یک خوشه است:



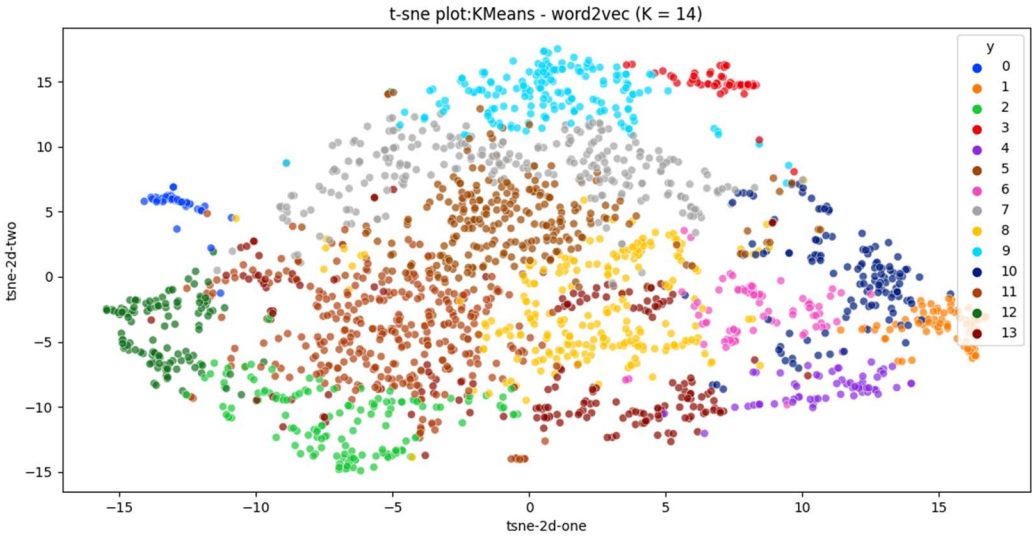


همانطور که از نمودارها مشخص است نمودار تبدیل Word2vec نمایش بهتری از خوشه‌بندی دارد.

همچنین داده‌ها را با استفاده از روش T-SNE[1] نیز کاهش بعد داده و رسم کردیم که در ادامه آمده است.



[1] T-Distributed Stochastic Neighbouring Entities



که همانطور که مشخص است کاهش بعد به روش T-SNE بهتر می‌تواند فاصله داده‌ها را در دو بعد نمایش دهد. و مشاهده می‌کنیم که تبدیل Word2vec فاصله‌ی داده‌ها را بهتر نمایش می‌دهد.

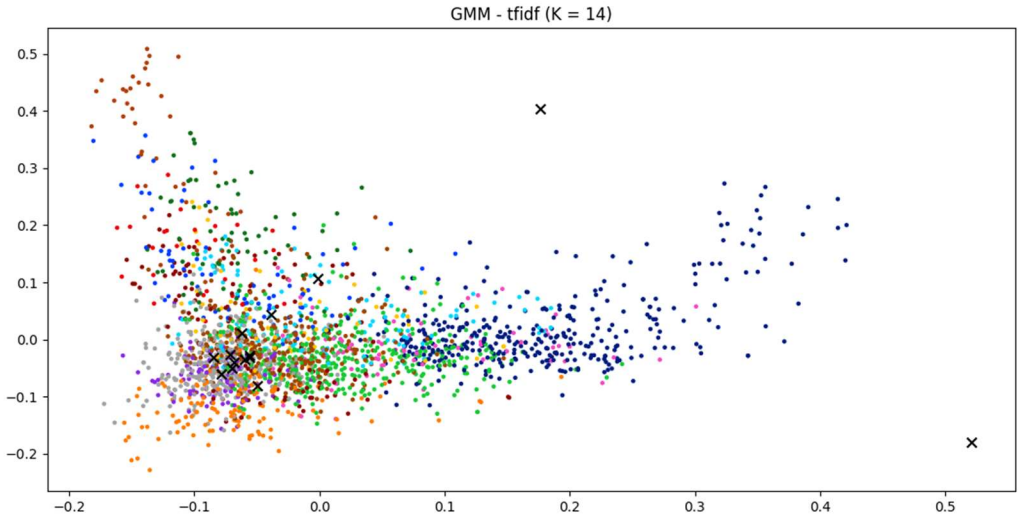
Gaussian Mixture Model .1.2

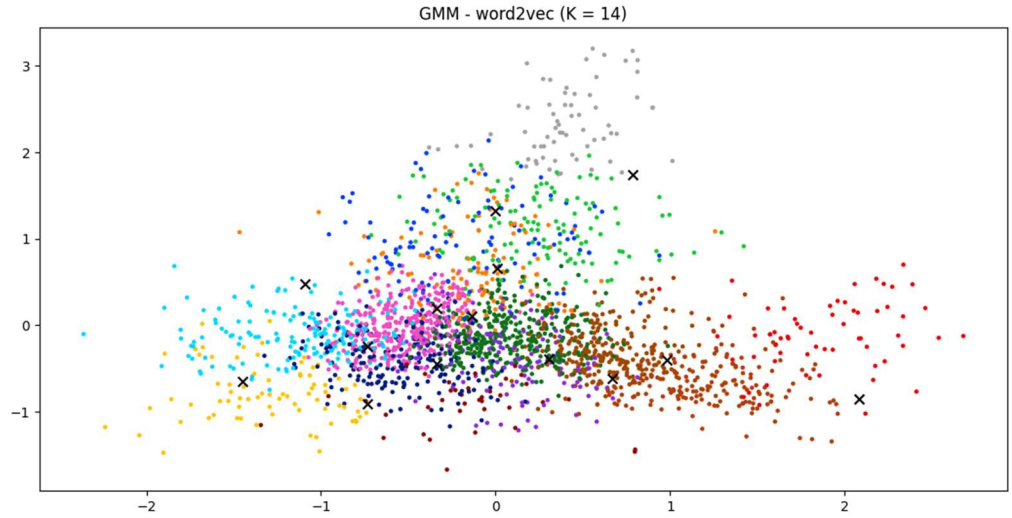
در این الگوریتم در هر مرحله به جای اختصاص هر داده به طور خاص به یک خوشه، برای آن را به یک احتمال به هر خوشه نسبت می‌دهیم و هدف ماکسیمم کردن لگاریتم تابع likelihood است. این تابع نیز با استفاده از تابع کتاب‌خانه‌ای آماده sklearn.mixture.GaussianMixture پیاده‌سازی شده است. برای ساختن مدل دو حالت full و diag با میانگین ۱۰ بار اجرا الگوریتم به ازای هر روش تست شد و حالت full با نتیجه بهتر انتخاب شد. نتیجه خوشه‌بندی با این الگوریتم نیز به ترتیب بالا به شرح زیر است:

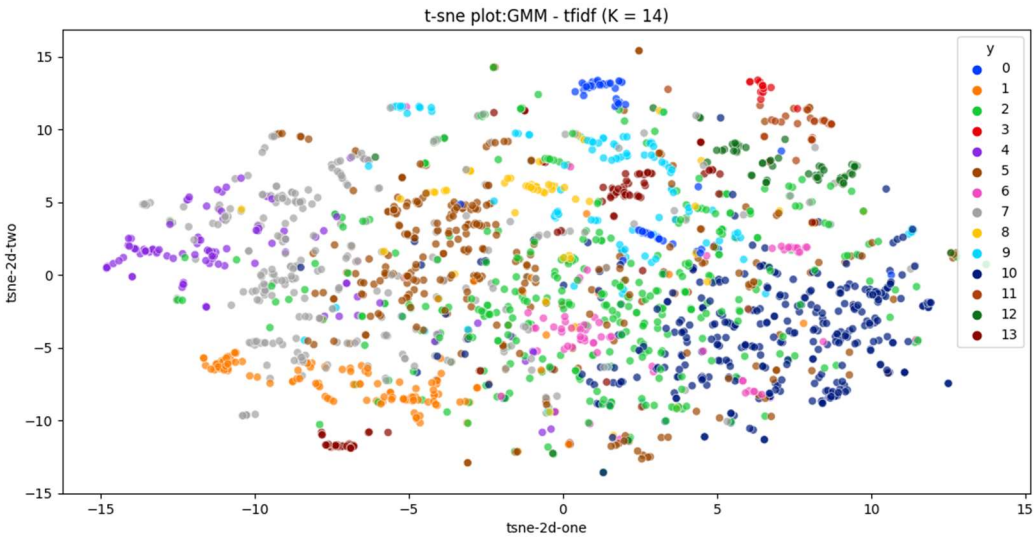
Output:

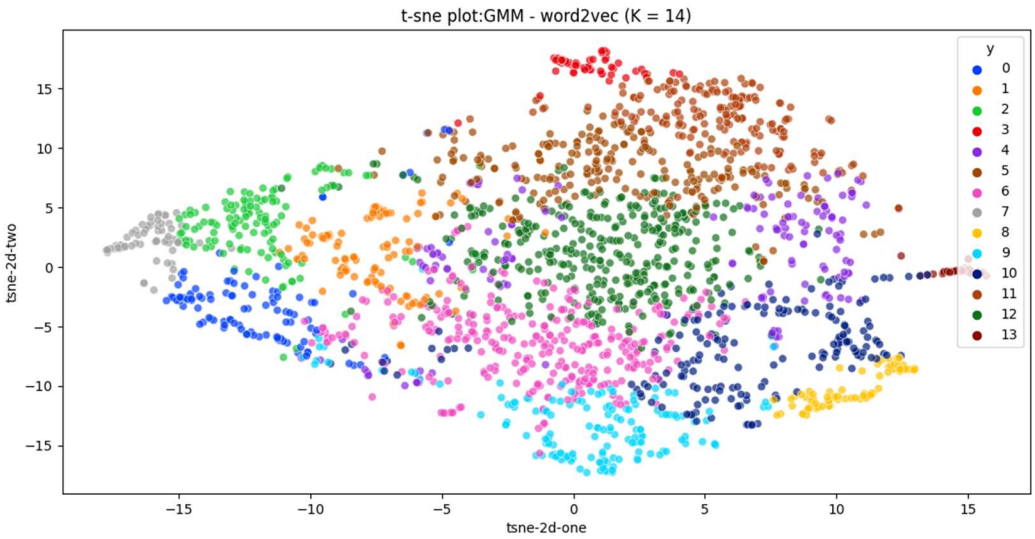
```
-> GMM clustering algorithm and tfidf vectorizer
   Purity score: 0.48978665456196097 , Rand Index score: 0.8143980856754248 ,
Adjusted Mutual Info score: 0.2642626182229385 , Adjusted Rand Index score:
0.1393713528890959
-> GMM clustering algorithm and word2vec vectorizer
   Purity score: 0.4298683613254653 , Rand Index score: 0.8223176800853267 ,
Adjusted Mutual Info score: 0.20937878908735819 , Adjusted Rand Index score:
0.11438128184510557
```

همچنین نمودارهای ۲ بعدی به ترتیب با روش‌های تبدیل به فضای برداری tfidf و Word2vec و کاهش ابعاد با روش‌های PCA و T-SNE در زیر آمده است.




GMM - word2vec (K = 14)


t-sne plot:GMM - tfidf (K = 14)
tsne-2d-one
tsne-2d-two
y
0
1
2
3
4
5
6
7
8
9
10
11
12
13


t-sne plot:GMM - word2vec (K = 14)
tsne-2d-one
tsne-2d-two
y
0
1
2
3
4
5
6
7
8
9
10
11
12
13

مانند قبل تبدیل Word2vec و روش T-SNE نمایش بهتری دارند. و دو روش تبدیل تقریبا عملکرد مشابهی دارند. tfidf کمی بهتر عمل کرده است. همچین در مقایسه با الگوریتم K-means تقریبا مشابه عمل کرده است. در کل باتوجه به اینکه انتخاب اولیه برای مراکز دسته رندوم است، هر کدام از این در دو روش ممکن از در بعضی از اجراها کمی بهتر از دیگری عمل کند. اما تفاوت قابل ملاحظه‌ای بین آن‌ها وجود ندارد.

1.3. Hierarchical Clustering

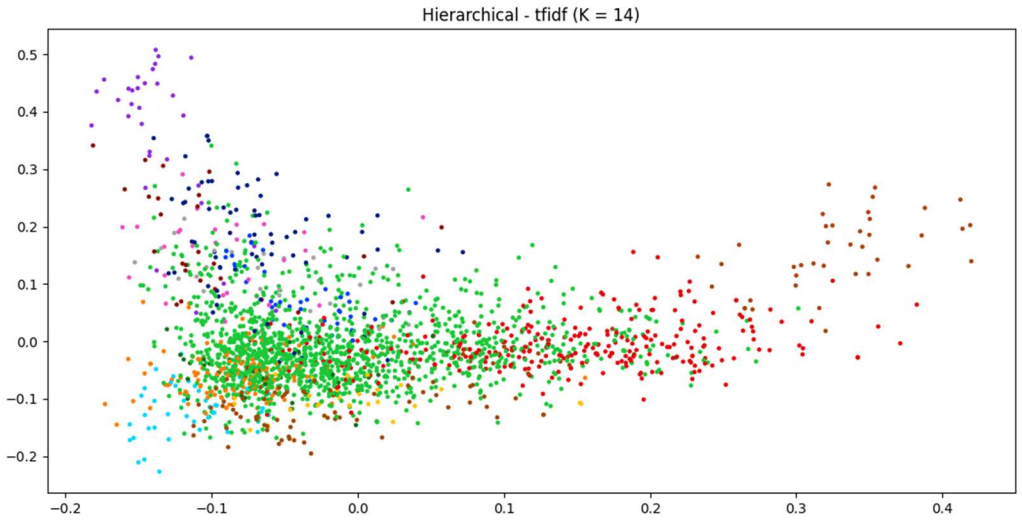
این روش نیز با استفاده از تابع کتاب‌خانه آماده *sklearn.cluster.AgglomerativeClustering* استفاده شده است. نتیجه اجرا به شرح زیر است:

Output:

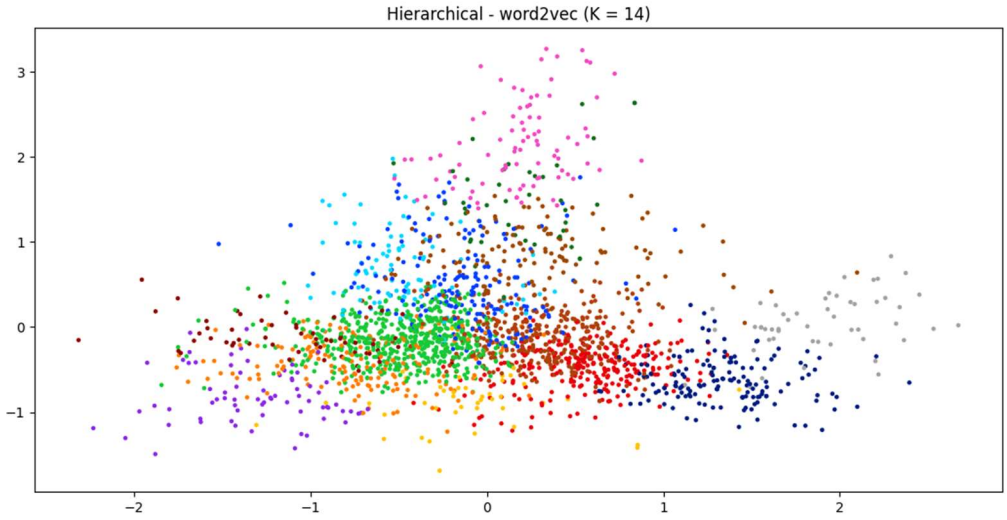
```
-> Hierarchical clustering algorithm and tfidf vectorizer
   Purity score: 0.40717203812982294 , Rand Index score: 0.5932344754881771 ,
Adjusted Mutual Info score: 0.23543734636260763 , Adjusted Rand Index score:
0.025816842403739486
-> Hierarchical clustering algorithm and word2vec vectorizer
   Purity score: 0.4475714934180663 , Rand Index score: 0.814617009337857 ,
Adjusted Mutual Info score: 0.2167958812505463 , Adjusted Rand Index score:
0.10706602398312193
```

تبدیل tfidf برای این الگوریتم به نسبت بد عمل می‌کند. اما نتیجه‌ی آن به ازای Word2vec تقریبا مشابه الگوریتم‌های قبلی است.

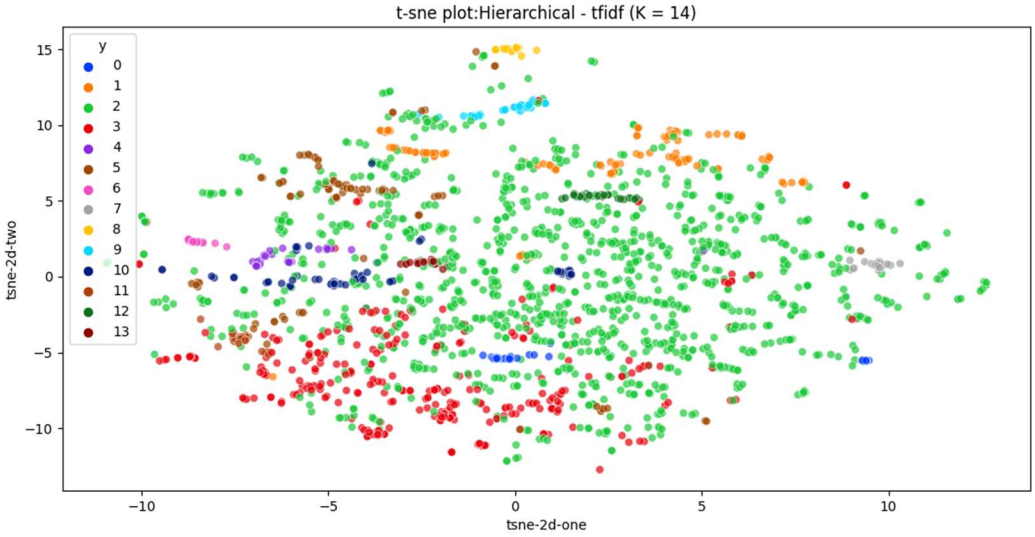
نمودارهای این قسمت به صورت زیر است. در این قسمت نمودار Dendrogram تا عمق حداکثر ۵ رسم شده است.

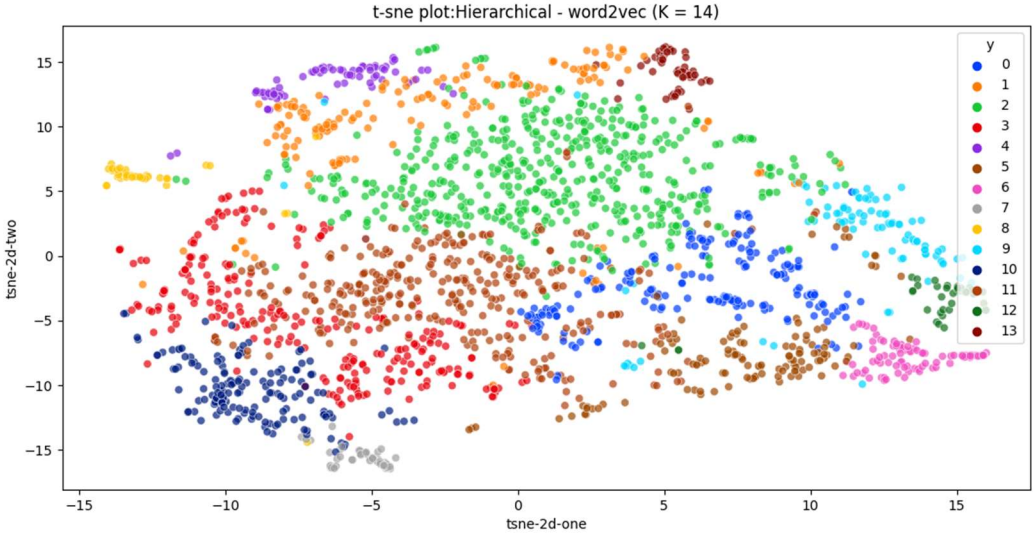


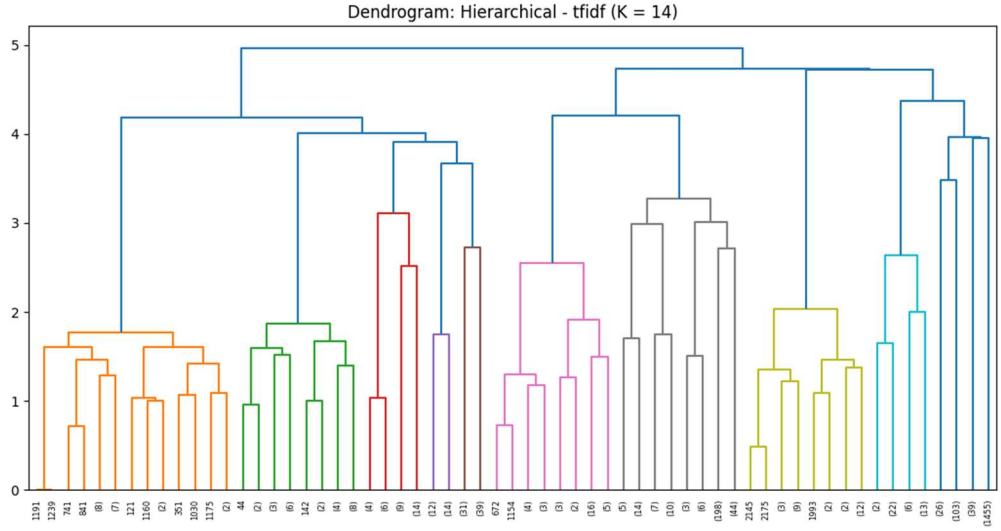
با توجه به شکل نیز می‌توان دید الگوریتم Hierarchical با استفاده از tfidf خوب عمل نمی‌کند.

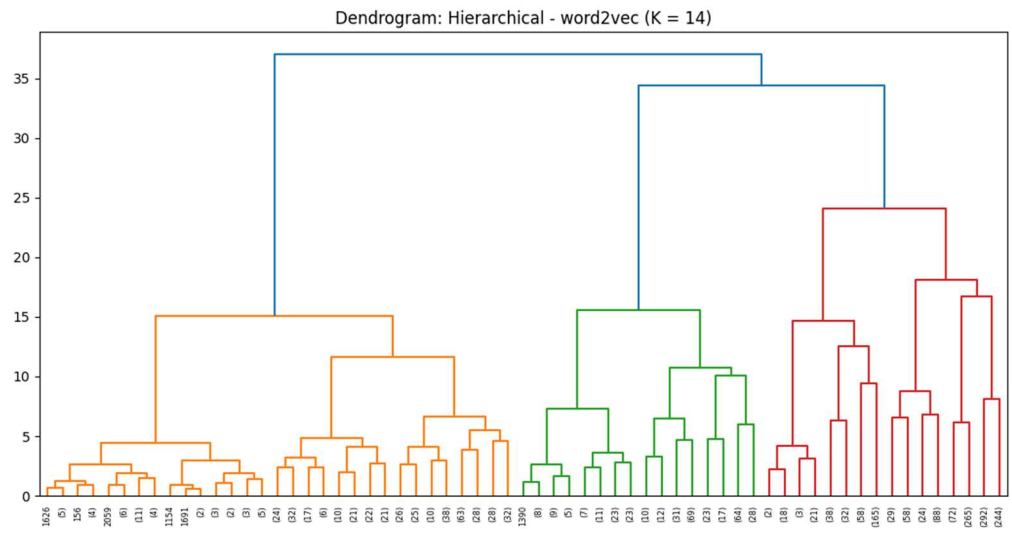


اما با استفاده از word2vec تقریبا مشابه قبل عمل می‌کند. در این الگوریتم تفاوت کار دو تبدیل مشخص می‌شود.




t-sne plot:Hierarchical - word2vec (K = 14)
tsne-2d-two
tsne-2d-one
y
0
1
2
3
4
5
6
7
8
9
10
11
12
13


Dendrogram: Hierarchical - tfidf (K = 14)


Dendrogram: Hierarchical - word2vec (K = 14)
35
30
25
20
15
10
5
0
1626
(5)
156
(4)
2059
(6)
(11)
(4)
1154
1691
(2)
(3)
(2)
(3)
(5)
(24)
(32)
(17)
(6)
(10)
(21)
(22)
(21)
(26)
(25)
(10)
(38)
(63)
(28)
(28)
(32)
1390
(8)
(9)
(5)
(7)
(11)
(23)
(23)
(10)
(12)
(31)
(69)
(23)
(17)
(64)
(28)
(2)
(18)
(3)
(21)
(38)
(32)
(58)
(165)
(29)
(58)
(24)
(88)
(72)
(265)
(292)
(244)

بخش ۲. پیاده‌سازی خزنده، واکشی اطلاعات مقالات

در این بخش با استفاده از selenium صفحه‌ها را crawl کردیم. برای واکشی هرکدام از element های مورد نظر (مثل title و abstract) از css selector مخصوصش که با inspect در صفحه به دست می‌آید استفاده می‌کنیم. برای reference ها فقط 10 تای اول لازم است، پس از واکشی از آنهایی که لینک دارند 10 تای اول را انتخاب می‌کنیم.

یک queue داریم که همه ی لینک‌هایی که می‌خواهیم واکشی کنیم را در آن قرار می‌دهیم. در ابتدا فقط 3 لینک درون start.txt را در queue می‌ریزیم. با واکشی هر صفحه 10 reference اول آن را به queue اضافه می کنیم.

همچنین، همه‌ی لینک‌ها را در یک لیست به نام cache می‌ریزیم تا بتوانیم تکراری‌ها را تشخیص دهیم. هر صفحه‌ی جدید با اعضای cache مقایسه می‌شود و اگر تکراری باشد وارد cache و queue نمی‌شود.

از طرفی در تابع crawl در یک حلقه ی وایل url ها را از queue پاپ می‌کنیم و آن‌ها را واکشی می‌کنیم. این کار را تا زمانی ادامه می‌دهیم که یا به 5000 صفحه رسیده باشیم یا queue خالی شده باشد.

با هر واکشی یک json از اطلاعات خواسته شده می‌گیریم و به لیست نهایی اضافه می‌کنیم. در پایان این لیست نهایی را در فایل papers.json می‌ریزیم.

در انتها به ازای هر الگوریتم و هر یک از روش‌های تبدیل به فضای برداری یک فایل اکسل در پوشه report ایجاد کردیم که نتیجه خوشه‌بندی را برای هر مستند گزارش می‌دهد. در ستون اول در این اکسل لینک مستندات و در ستون دوم آن شماره‌ی خوشه‌ای که به آن نسبت داده شده است قرار دارد.

بخش ۳. PageRank

از فایل papers.json که در بخش قبل ساختیم، لیستی از jsonهای ذخیره شده رو لود می‌کنیم. با استفاده از networkx یک گراف می‌سازیم که در آن از هر مستند به reference هایش یالی کشیده شده است. سپس با استفاده از آلفایی که در ورودی می‌گیریم و گراف ساخته شده، PageRank ها را حساب می کنیم. ۱۰ مقاله با بالاترین PageRank را دارند به صورت زیر است، مقدار آلفا ۰,۱ وارد شده است: (متاسفانه فرایند بخش قبلی یعنی خزش و واکشی اطلاعات خیلی زمان‌بر بود و خدودا ۱۲ ساعت برای ۳۰۰۰ مقاله زمان برده است. برای اینکه کمتر مشول تاخیر شویم این بخش را ۳۳۰۰ مقاله که تا به این لحظه واکشی شده است اجرا می‌کنیم. نتیجه اجرا روی ۵۰۰۰ مقاله را هر وقت آماده شد در کوئرا بارگذاری می‌کنیم).

```
(('Maximum likelihood from incomplete data via the EM algorithm',
'2049633694'), 0.000809068341199982),
(('ImageNet classification with deep convolutional neural networks',
'2618530766'), 0.000715086535601176),
(('Clinical features of patients infected with 2019 novel coronavirus in
Wuhan, China', '3001118548'), 0.0007049896686674888),
(('Learning internal representations by error propagation', '2154642048'),
0.0007026779252067686),
(('Pattern classification and scene analysis', '3017143921'),
0.0006973218099925366),
(('Gradient-based learning applied to document recognition', '2310919327'),
0.000671888757806819),
(('Stochastic relaxation, Gibbs distributions and the Bayesian restoration of
images*', '1997063559'), 0.0006029529189872005),
(('Isolation of a Novel Coronavirus from a Man with Pneumonia in Saudi
Arabia', '2166867592'), 0.0005963551589011609),
(('A Novel Coronavirus from Patients with Pneumonia in China, 2019.',
'3001897055'), 0.0005932775732974887),
(('The Nature of Statistical Learning Theory', '2156909104'),
0.0005801198418044535),
(('Optimization by simulated annealing', '2581275558'),
0.0005715717567736365),
(('Coronavirus as a possible cause of severe acute respiratory syndrome',
'2025170735'), 0.0005711230068671696),
(('Early Transmission Dynamics in Wuhan, China, of Novel Coronavirus-Infected
Pneumonia.', '3003668884'), 0.0005647474055120065),
(('Identification of a novel coronavirus in patients with severe acute
respiratory syndrome.', '2132260239'), 0.0005609005798187619),
(('APACHE II: a severity of disease classification system.', '2107978811'),
0.0005557159427473803),
(('A novel coronavirus associated with severe acute respiratory syndrome.',
'2104548316'), 0.0005512355338958634),
(('Long short-term memory', '2064675550'), 0.0005414789978910803)
```