# 

# **BAB IV**

# **HASIL DAN PEMBAHASAN**

## Hasil Pengumpulan Data

Data yang digunakan dalam penelitian “*Text Mining untuk* pengelompokkan judul Penelitian Dosen menggunakan metode *K-Medoids* dengan *Cosine similarity*“ yaitu judul penelitian dosen Universitas Ahmad Dahlan, didapat dari LPPM Universitas Ahmad Dahlan.

Data berupa *file excel* dengan nama “*DATA PENELITIAN DOSEN.xls”.* Data judul penelitian dosen berupa tabel dengan data sebanyak 517 judul penelitian dan dilengkapi dengan atribut judul. Data yang digunakan dalam waktu 5 tahun terakhir (2012-2017) , yang akan dikelompokkan pada penelitian ini yaitu pada atribut Judul Penelitian.

## Tahapan Text Mining

1. ***Load* Data**

Data yang digunakan yaitu Judul Penelitian dosen, setiap judul dianggap dokumen dan diberikan ID pada setiap dokumen seperti ditunjukan pada Tabel 4.1.

Tabel 4.1. Load data

|  |  |
| --- | --- |
| **ID** | **JUDUL** |
| D0 | Penerapan antar muka bahasa alami dalam pencarian informasi skripsi pada suatu program studi |
| D1 | Sistem reminder aktifitas akademik dosen |
| D2 | Perancangan Sistem Test Komputerisasi dan Pendukung Keputusan Penerimaan Pegawai |
| D3 | Pengolahan Bahasa Alami pada Agen Cerdas Alat Bantu Konsultasi Zakat |

|  |  |
| --- | --- |
| D4 | Perancangan dan implementasi system pengenalan jenis kulit hewan unutk kerajinan kulit berbasis system cerdas |
| D5 | Pengembangan Template Media Pembelajaran Berbasis Flash |
| D6 | Membangun system informasi eksekutif (SIE) dengan menggunakan pendekatan system cerdas |
| D7 | Perancangan visualisasi informasi untuk system evaluasi guru |
| D8 | Pengembangan Sistem Penilaian Kerja Dosen dengan Aspek IKD dan SKP 2014 Berbasis Online |
| D9 | Perancangan animasi wayang pendidikan sebagai pengembangan media edukasi sekolah |

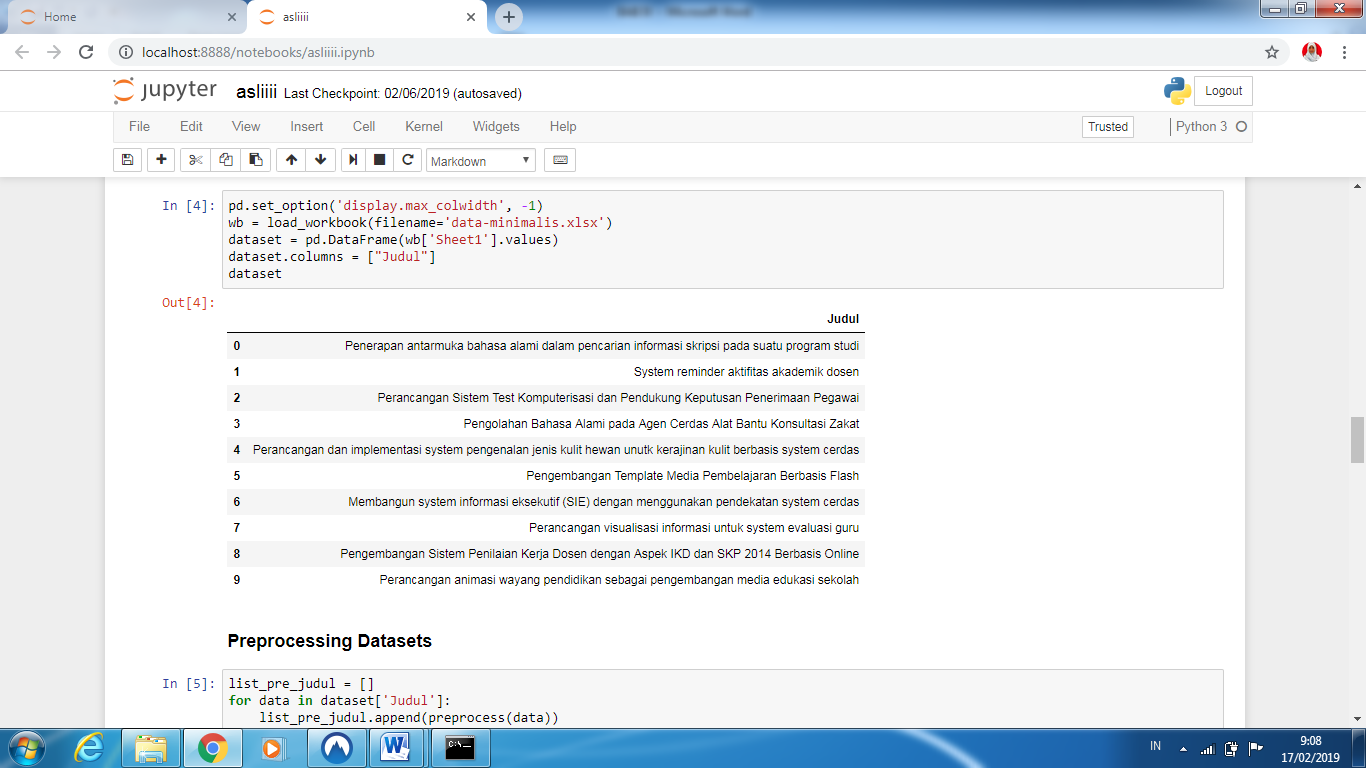
Tahap ini dilakukan untuk melakukan *load* data judul penelitian dosen dalam bentuk excel ke python, Listing program dapat dilihat pada Listing 4.1 *Load* data dan Gambar 4.1 hasil *Load* data.

|  |  |
| --- | --- |
| 1 | pd.set\_option('display.max\_colwidth', -1) |
| 2 | wb = load\_workbook(filename='data-minimalis.xlsx') |
| 3 | dataset = pd.DataFrame(wb['Sheet1'].values) |
| 4 | dataset.columns = ["Judul"] |
| 5 | Dataset |

Listing 4.1

Keterangan :

1. Baris 2 melakukan load data dengan nama wb =load\_workbook
2. Baris 3-4 variabel dataset dengan sheet 1
3. Baris 4 nama dataset kolom judul

  
Gambar 4.1 Hasil load data

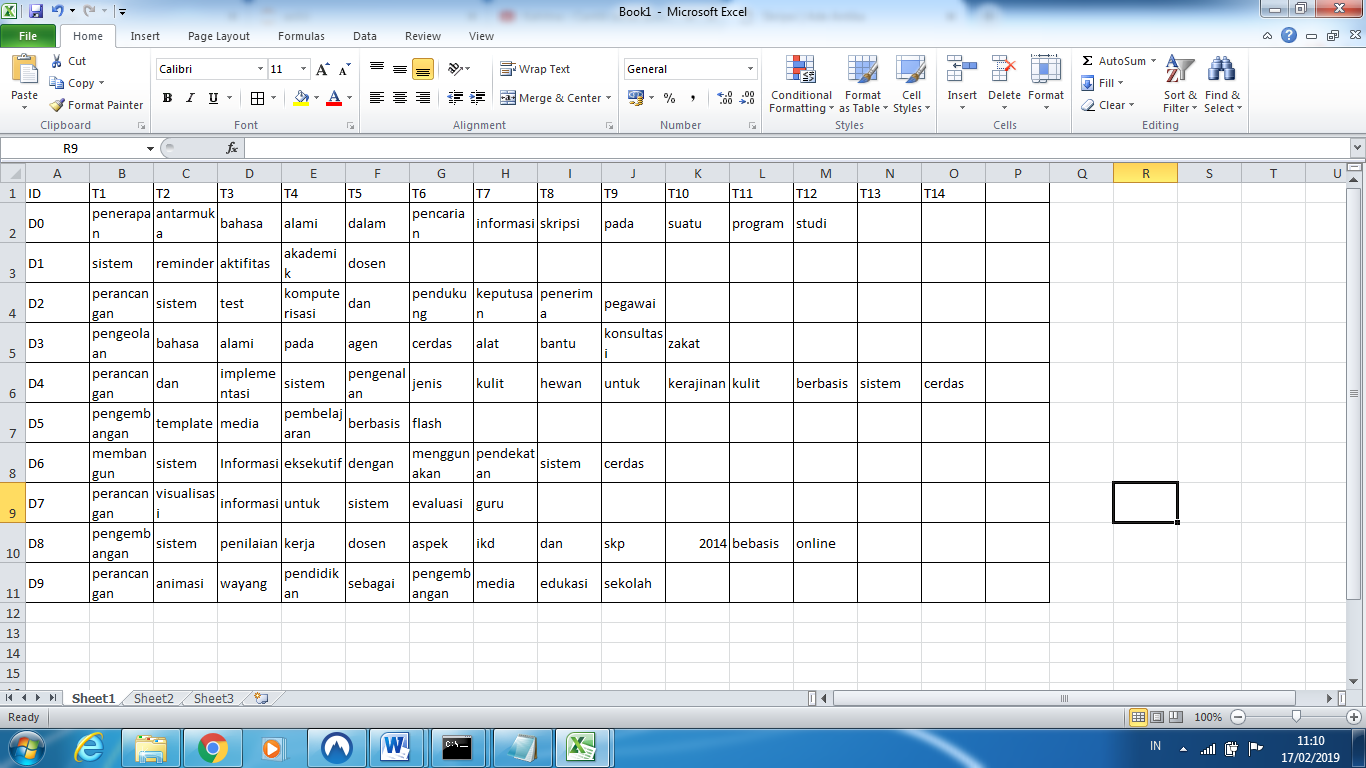
1. **Preprocessing**

Preprocessing merupakan tahapan dari teks mining yang tahapan nya terdiri dari *tokenisasi, filtering dan stemming.*

**4.3.1 Tokenisasi**

Tokenisasiadalah proses pemecahan kalimat menjadi kata berdasarkan tiap judul, tiap kata akan di beri ID seperti dilihat pada Tabel 4.2.

Tabel 4.2. *tokenisasi*



* + 1. ***Text Filtering***

Setelah proses tokenisasi *s*elanjutnya yaitu proses filterisasi, dimana fungsi filterisasi adalah menghapus kata penghubung pada setiap dokumen. Hasil proses filterisasi ditunjukkan pada Tabel 4.3

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 |
| D0 | penerapan | antarmuka | bahasa | alami | pencarian | informasi | skripsi | progam | Studi |  |  |  |
| D1 | sistem | reminder | aktifitas | akademik | dosen |  |  |  |  |  |  |  |
| D2 | perancangan | sistem | test | komputerisasi | pendukung | Keputusan | penerima | pegawai |  |  |  |  |
| D3 | pengeolaan | bahasa | alami | agen | cerdas | Alat | bantu | konsultasi | zakat |  |  |  |
| D4 | perancangan | implementasi | sistem | pengenalan | jenis | kulit | hewan | kerajinan | kulit | berbasis | Sistem | cerdas |
| D5 | pengembangan | template | media | pembelajaran | berbasis | flash |  |  |  |  |  |  |
| D6 | membangun | sistem | Informasi | eksekutif | menggunakan | Pendekatan | sistem | cerdas |  |  |  |  |
| D7 | perancangan | visualisasi | informasi | sistem | evaluasi | guru |  |  |  |  |  |  |
| D8 | pengembangan | sistem | penilaian | kerja | dosen | aspek | ikd | skp | 2014 | berbasis | online |  |
| D9 | perancangan | animasi | wayang | pendidikan | pengembangan | media | edukasi | sekolah |  |  |  |  |

Tabel 4.3. *Filtering*

* + 1. ***Stemming***

Setelah melakukan proses *filtering* proses selanjutnya adalah *stemming. Stemming* yaitu proses untuk menghilangkan imbuhan pada setiap kata agar menjadi kata dasar. Hasil dari proses *stemming* di tunjukkan pada Tabel 4.4*.*

Table 4.4 *Stemming*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 |
| D0 | terap | antarmuka | bahasa | alami | cari | informasi | skripsi | program | studi |  |  |
| D1 | system | reminder | aktifitas | akademik | dosen |  |  |  |  |  |  |
| D2 | rancang | sistem | Test | komputerisasi | dukung | putus | terima | pegawai |  |  |  |
| D3 | kelola | bahasa | alami | agen | cerdas | alat | bantu | konsultasi | zakat |  |  |
| D4 | rancang | implementasi | sistem | kenal | jenis | kulit | hewan | rajin | basis | cerdas |  |
| D5 | kembang | Template | media | Ajar | basis | flash |  |  |  |  |  |
| D6 | bangun | Sistem | informasi | eksekutif | guna | dekat | sistem | cerdas |  |  |  |
| D7 | rancang | visualisasi | informasi | sistem | evaluasi | guru |  |  |  |  |  |
| D8 | kembang | sistem | nilai | kerja | dosen | aspek | Ikd | skp | 2014 | basis | online |
| D9 | rancang | animasi | wayang | didik | kembang | media | edukasi | sekolah |  |  |  |

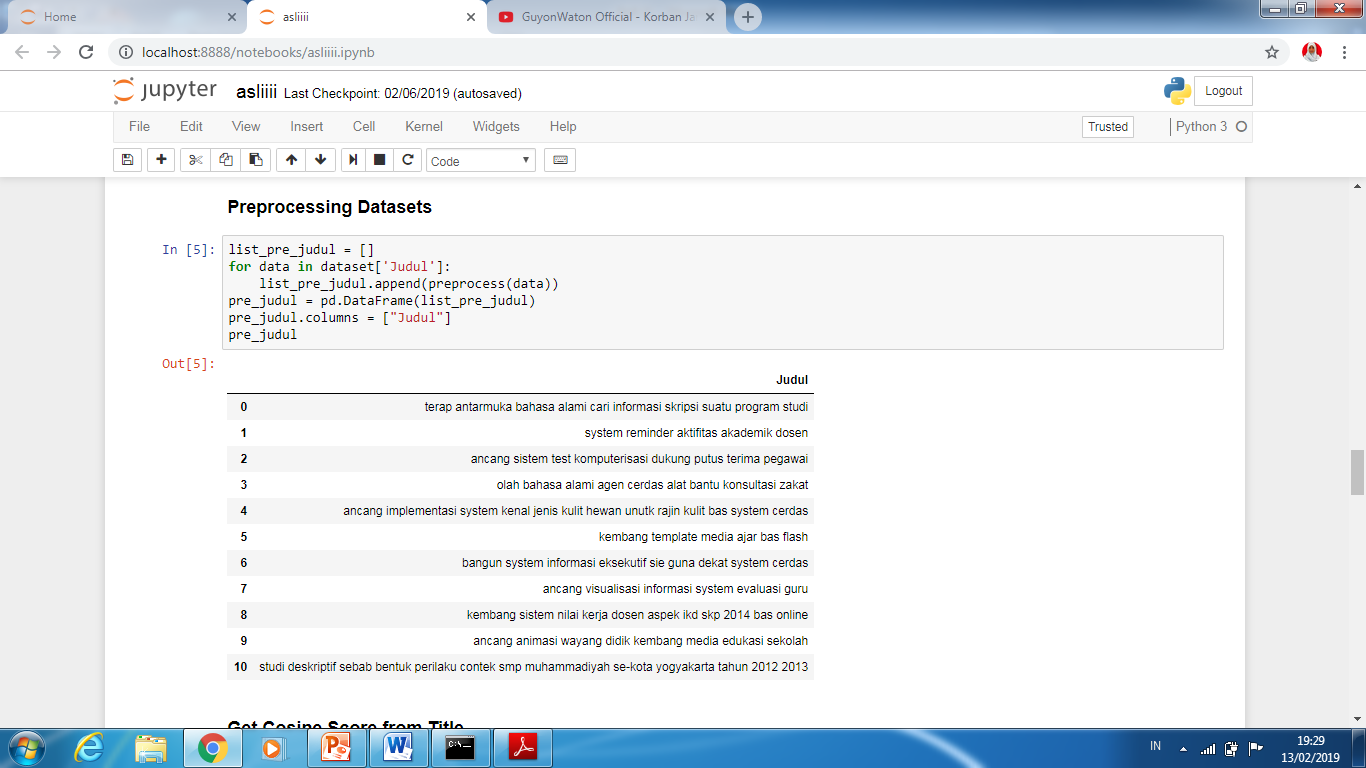
Setelah melakukan proses hasil load data dari excel yang ditunjukkan pada Gambar 4.1 hasil Load data, maka tahapan selanjutnya adalah melakukan proses *preprocessing.* *Preprocessing* merupakan tahapan dari pada teks *Mining* yang mengubah bentuk kalimat yang tidak terstruktur menjadi terstruktur proses Listing pada preprocessing ditunjukkan pada Listing 4.2 dan Hasil *Preprocessing* pada Gambar 4.3 Hasil *Preprocessing*

|  |  |
| --- | --- |
| 1 | def preprocess(text): |
| 2 | text = text.lower() |
| 3 | text\_clean = remover.remove(text) #fungsi hapus stopword |
| 4 | text\_stem = stemmer.stem(text\_clean) |
| 5 | return text\_stem |
| 6 | list\_pre\_judul = [] |
| 7 | for data in dataset['Judul']: |
| 8 | list\_pre\_judul.append(preprocess(data)) |
| 9 | pre\_judul = pd.DataFrame(list\_pre\_judul) |
| 10 | pre\_judul.columns = ["Judul"] |
| 11 | pre\_judul |

Listing 4.2 *preprocessing*

Keterangan :

1. Baris 1, fungsi untuk preprocessing text bahasa indonesia
2. Baris 2 , Mengubah text menjadi huruf kecil
3. Baris 3, fungsi hapus stopword
4. Baris 4, fungsi untuk stemming
5. Baris 6, Membuat list untuk menampung judul
6. Baris 7 -8 proses preprocessing dataset

  
Gambar 4.2 hasil *preprocessing*

***4.4. Cosine Similarity***

Poses *cosine similarity* digunakan untuk mengukur mengukur kesamaan antara dua vektor dengan mengambil kosinus sudut yang dibuat dua vektor pada ruang produk titik mereka.

D0 = Penerapan antar muka bahasa alami dalam pencarian informasi skripsi pada suatU program studi  
D3 = Pengolahan Bahasa Alami pada Agen Cerdas Alat Bantu Konsultasi Zakat.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | T1 | T2 | T3 | T4 | T5 | T6 | T7 |  | T8 | T9 |  |  |  |  |  |  |  |
| D0 | terap | antarmuka | bahasa | Alami | cari | informasi | skripsi | suatu | program | studi | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
| D3 |  |  | bahasa | alami |  |  |  |  |  |  | olah | agen | cerdas | alat | bantu | konsultasi | zakat |
|  | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabel4.5 *Cosine Similarity*

Tabel 4. 6 matrix cosine similarity

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 |
| D0 | 1.000 | 0 | 0 | 0.211 | 0 | 0 | 0.095 | 0.129 | 0 | 0 |
| D1 | 0 | 1.000 | 0 | 0 | 0.217 | 0 | 0.270 | 0.183 | 0.135 | 0 |
| D2 | 0 | 0 | 1.000 | 0 | 0.086 | 0 | 0 | 0.143 | 0.107 | 0.125 |
| D3 | 0.211 | 0 | 0 | 1.000 | 0.081 | 0 | 0.101 | 0 | 0 | 0 |
| D4 | 0 | 0.217 | 0.086 | 0.081 | 1.000 | 0.099 | 0.366 | 0.297 | 0.073 | 0.086 |
| D5 | 0 | 0 | 0 | 0 | 0.099 | 1.000 | 0 | 0 | 0.246 | 0.289 |
| D6 | 0.095 | 0.270 | 0 | 0.101 | 0.366 | 0 | 1.000 | 0.369 | 0 | 0 |
| D7 | 0.129 | 0.183 | 0.144 | 0 | 0.297 | 0 | 0.369 | 1.000 | 0 | 0.144 |
| D8 | 0 | 0.135 | 0.107 | 0 | 0.073 | 0.246 | 0 | 0 | 1.000 | 0.107 |
| D9 | 0 | 0 | 0.125 | 0 | 0.086 | 0.289 | 0 | 0.144 | 0.107 | 1.000 |
| **∑** | 1.435 | 1.805 | 1.462 | 1.393 | 2,305 | 1.634 | 2.199 | 2.265 | 1.668 | 1.751 |

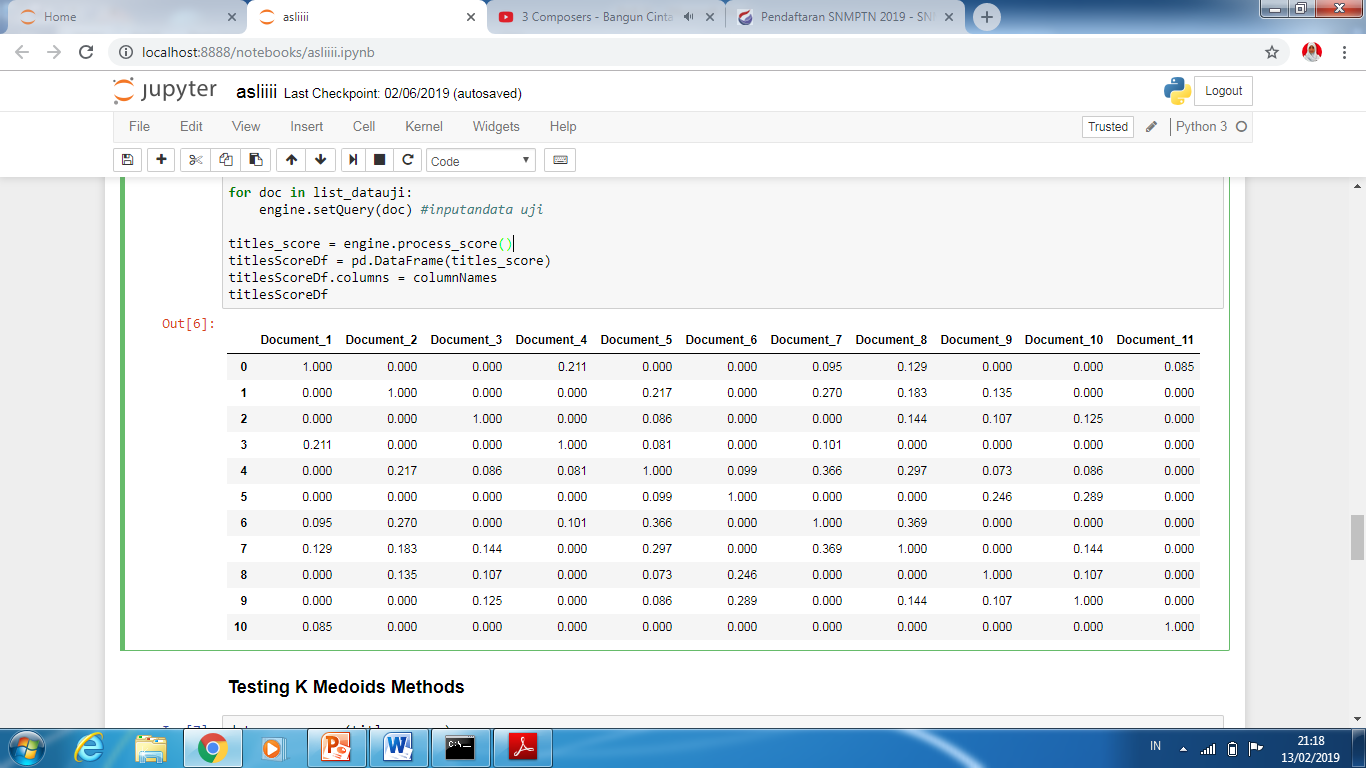
Setelah melakukan proses *prepocessing* yang ditunjukkan pada Gambar 4.2 hasil *preprocessing*, maka tahapan selanjutnya adalah menghitung nilai dari pada kedekatan antar dokumen, perhitungan *Cosine Similarity* secara manual dapat dilihat pada Tabel 4.5 contoh dokumen yang dihitung secara manual adalah menggunakan D0 dan D3 dengan hasil 0,211 dan *matrix* yang terbentuk dari hasil perhitungan pada Tabel 4.6 *matrix Cosine similarity.* Proses Listing dapat dilihat pada Listing 4.3 *Cosine Similarity* dan Gambar 4.3 Hasil *Cosine Similarity.* Dari perhitungan manual pada *Cosine* yang ditunjukkan pada Gambar 4.6 *matrix Cosine Similarity,* maka untuk hasil perhitungan sistem dapat dilihat pada Gambar 4.3 Hasil *Cosine Similarity.* Perhitungan sistem pada dokumen D0 dan D3 adalah 0,211.

|  |  |
| --- | --- |
| 1 | engine = Engine() |
| 2 | list\_dokumen = [str(x) for x in pre\_judul['Judul']] |
| 3 | list\_datauji = [str(x) for x in pre\_judul['Judul']] |
| 4 | columnNames = [] |
| 5 | for i, doc in enumerate(list\_dokumen): |
| 6 | engine.addDocument(doc) |
| 7 | columnNames.append("Document\_{}".format(i+1)) |
| 8 | for doc in list\_datauji: |
| 9 | engine.setQuery(doc) #inputandata uji |
| 10 | titles\_score = engine.process\_score() |
| 11 | titlesScoreDf = pd.DataFrame(titles\_score) |
| 12 | titlesScoreDf.columns = columnNames |
| 13 | titlesScoreDf |

Listing 4.3 *Cosine Similarity*

Keterangan :

1. Baris 1, membuat fungsi construct agar bisa di panggil di semua fungsi dalam class engine.
2. Baris 2 -3 list data dokumen dengan list data uji
3. Baris 5, melakukan perulangan untuk add document dengan fungsi engine
4. Baris 8, melakukan list data uji dengan fungsi engine.set query diubah dalam bentuk array
5. Baris 10 melakukan proses engine prosess\_score untuk nilai *cosine*

  
Gambar 4.3 Hasil *Cosine Similarity*

* 1. ***K-Medoids***

Data percobaan pertama dengan 6 cluster mengacu pada bidang penelitian dosen, 6 bidang tersebut yaitu Teknologi Informasi, Sosial dan Pendidikan, Ekonomi, Agama, Kesehatan dan Obat, dan Hukum. 6 cluster tersebut diambil dari sistem yang mengambil data secara random yaitu titik pusat dokumen 2 (D2) dokumen 0 (D0), dokumen 5 (D5), dokumen 6(D6), dokumen 8(D8), dokumen 9(D9). Dengan menggunakan metode *K-Medoids* untuk menentukan keanggotaan seluruh dokumen terhadap medoid D2,D0,D5,D6,D8 dan D9. Bandingkan seluruh dokumen dengan nilai *cosine* D2,D0,D5,D6,D8 dan D9, jika jarak dokumen terhadap D2 lebih besar dari pada jarak terhadap D0 maka dokumen tersebut masuk dalam *cluster* 0 (medoid D2), jika sebaliknya maka masuk dalam *cluster* 1 (medoid D0) ditunjukkan pada Tabel 4.7.

Tabel 4. 7 medoid tahap 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | D2 | D0 | D5 | D6 | D8 | D9 | Cluster |
| D0 | 0 | 1.000 | 0 | 0.095 | 0 | 0 | 1 |
| D1 | 0 | 0 | 0 | 0.270 | 0.135 | 0 | 3 |
| D2 | 1.000 | 0 | 0 | 0 | 0.107 | 0.125 | 0 |
| D3 | 0 | 0.211 | 0 | 0.101 | 0 | 0 | 1 |
| D4 | 0.086 | 0 | 0.099 | 0.366 | 0.073 | 0.086 | 3 |
| D5 | 0 | 0 | 1.000 | 0 | 0.246 | 0.289 | 2 |
| D6 | 0 | 0.095 | 0 | 1.000 | 0 | 0 | 3 |
| D7 | 0.144 | 0.129 | 0 | 0.369 | 0 | 0.144 | 3 |
| D8 | 0.107 | 0 | 0.246 | 0 | 1.000 | 0.107 | 4 |
| D9 | 0.125 | 0 | 0.289 | 0 | 0.107 | 1.000 | 5 |
|  | 1.462 | 1.435 | 1.634 | 2.199 | 1.668 | 1.751 |  |

Setelah itu menentukan proses cluster, pisahkan dokumen yang termasuk dalam *cluster 0* (titik pusat D2), *cluster* 1 (titik pusat D0), cluster 2(titik pusat D5), cluster 3(titik pusat D6) cluster 4(titik pusat D8) cluster 5(titik pusat D9). nilai yang masuk sebagai cluster adalah nilai yang mendekati 1. Agar lebih terlihat kelompok dokumen yang sudah terbentuk. Dari hasil pemisahan kelompok dokumen ditunjukkan pada Tabel 4.8.

Tabel 4. 8 Cluster

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cluster 0** | **Cluster 1** | **Cluster 2** | **Cluster 3** | **Cluster 4** | **Cluster 5** |
| Titikpusat D2  D2 | Titik pusat D0  D3 | Titik pusat D5  D5 | Titik pusat D6  D4  D7 | Titik pusat D8  D8 | Titik pusat D9  D9 |

Langkah selanjutnya *K-Medoids* akan membandingkan titik pusat pada *cluster 0* terlebih dahulu dengan dokumen yang menjadi anggota pada *cluster* 0.

1. *Cluster* 0 iterasi 1 : jumlahkan jarak data yang berada pada medoid ke -1 dan ke -2, jika jumlah medoid ke 1 lebih besar dari pada jumlah jarak medoid ke -2 maka medoid pada cluster 0 berubah menjadi data ke -2, namun jika sebaliknya jumlah medoid ke -1 lebih kecil dari pada jumlah medoid ke -2 maka medoid tetap. Pada cluster 0 titik pusat nya yaitu D2 dan anggota nya sama D2, maka titik pusat tidak berubah tetap pada D2. Dapat dilihat pada tabel 4.9 Tabel hasil cluster 0.

Tabel 4. 9 cluster 0 iterasi 1

|  |  |  |
| --- | --- | --- |
| ID | Jarak medoid D2 | Jarak medoid D2 |
| D0 | 0 | 0 |
| D1 | 0 | 0 |
| D2 | 1.000 | 1.000 |
| D3 | 0 | 0 |
| D4 | 0.086 | 0.086 |
| D5 | 0 | 0 |
| D6 | 0 | 0 |
| D7 | 0.144 | 0.144 |
| D8 | 0.107 | 0.107 |
| D9 | 0.125 | 0.125 |
| ∑ | 1.462 | 1.462 |

Perhitungan medoid *cluster* 0 iterasi 1 = D2= D2, maka medoid atau titik pusat pada *Cluster* 0 tetap pada D2 sebagai titik pusat. Dapat dilihat pada Tabel 4.10 titik pusat cluster 0

Tabel 4. 10 titik pusat cluster 0

|  |
| --- |
| **Cluster 0** |
| Titik pusat = D2  D2 |

1. Cluster 1 iterasi 1 dengan cara yang sama seperti point a bandingkan jumlah nilai cosine tiap dokumen bandingkan titik pusat D0 dengan D3. Dapat dilihat pada Tabel 4.11

Tabel 4. 11 cluster 1 iterasi 1

|  |  |  |
| --- | --- | --- |
| ID | Jarak medoid D0 | Jarak medoid D3 |
| D0 | 1.000 | 0.211 |
| D1 | 0 | 0 |
| D2 | 0 | 0 |
| D3 | 0.211 | 1.000 |
| D4 | 0 | 0.081 |
| D5 | 0 | 0 |
| D6 | 0.095 | 0.101 |
| D7 | 0.129 | 0 |
| D8 | 0 | 0 |
| D9 | 0 | 0 |
| ∑ | 1.435 | 1.393 |

Perhitungan medoid cluster 1 iterasi 1 D0 > D3 , maka medoid atau titik pusat cluster 1 berubah menjadi D3. Ditunjukkan pada tabel 4.12 Tabel hasil cluster 1

Tabel 4. 12 titik pusat cluster 1

|  |
| --- |
| **Cluster 1** |
| Titik pusat = D3  D0 |

1. Cluster 2 iterasi 1 dengan cara yang sama seperti point a bandingkan jumlah nilai cosine tiap dokumen pada cluster 2 untuk mencari titik pusat pada *cluster* 2, bandingkan titik pusat D5 dengan D5 .karena titik pusat dan anggota cluster sama maka titik pusat tetap pada D5. Dapat dilihat pada tabel 4.13 Tabel hasil cluster 2.

Tabel 4. 13 titik pusat cluster 2

|  |
| --- |
| **Cluster 2** |
| Titik pusat = D5  D5 |

1. Cluster 3 iterasi 1 : Membandingkan jumlah nilai cosine tiap dokumen pada cluster 3 untuk mecari titik pusat pada cluster 3, bandingkan titik pusat D6 dengan D4 terlebih dahulu ditunjukkan pada Tabel 4.14

Tabel 4. 14 cluster 3 iterasi 1

|  |  |  |
| --- | --- | --- |
| ID | Jarak medoid D6 | Jarak medoid D4 |
| D0 | 0.095 | 0 |
| D1 | 0.270 | 0.217 |
| D2 | 0 | 0.086 |
| D3 | 0.101 | 0.081 |
| D4 | 0.366 | 1.000 |
| D5 | 0 | 0.099 |
| D6 | 1.000 | 0.366 |
| D7 | 0.369 | 0.297 |
| D8 | 0 | 0.073 |
| D9 | 0 | 0.086 |
| ∑ | 2.199 | 2,305 |

Perhitungan medoid cluster 3 iterasi 1 D6 < D4 , maka medoid atau titik pusat cluster 3 tetap D6 sebagai titik pusat. Selanjutnya bandingkan D6 dengan D7 pada Tabel 4.15

Tabel 4.15 cluster 3 iterasi 2

|  |  |  |
| --- | --- | --- |
| ID | Jarak medoid D6 | Jarak medoid D7 |
| D0 | 0.095 | 0.129 |
| D1 | 0.270 | 0.183 |
| D2 | 0 | 0.143 |
| D3 | 0.101 | 0 |
| D4 | 0.366 | 0.297 |
| D5 | 0 | 0 |
| D6 | 1.000 | 0.369 |
| D7 | 0.369 | 1.000 |
| D8 | 0 | 0 |
| D9 | 0 | 0.144 |
| ∑ | 2.199 | 2.265 |

Perhitungan medoid cluster 3 iterasi 2 = D6< D7 , maka medoid atau titik pusat pada cluster 3 tetap D6 sebagai titik pusat. Ditunjukkan pada Tabel 4.16

Tabel 4. 16 titik pusat cluster 3

|  |
| --- |
| **Cluster 3** |
| Titik pusat = D6  D4  D7 |

1. Cluster 4 iterasi 1 dengan cara yang sama seperti point a bandingkan jumlah nilai cosine tiap dokumen pada cluster 4 untuk mencari titik pusat pada *cluster* 4, bandingkan titik pusat D8 dengan D8.karena titik pusat dan anggota cluster sama maka titik pusat tetap pada D8. Dapat dilihat pada tabel 4.17 Tabel hasil cluster 4

Tabel 4. 17 titik pusat cluster 4

|  |
| --- |
| **Cluster 4** |
| Titik pusat = D8  D8 |

1. Cluster 5 iterasi 1 dengan cara yang sama seperti point a bandingkan jumlah nilai cosine tiap dokumen pada cluster 5 untuk mencari titik pusat pada *cluster* 5, bandingkan titik pusat D9 dengan D9 .karena titik pusat dan anggota cluster sama maka titik pusat tetap pada D9. Dapat dilihat pada tabel 4.18 Tabel hasil cluster 5.

Tabel 4. 18 titik pusat Cluster 5

|  |
| --- |
| **Cluster 5** |
| Titik pusat = D9  D9 |

Kemudian dibandingkan kembali anggota tiap cluster untuk melihat apakah anggota tetap sama atau berubah. Pada Tabel 4.19 cluster.

Tabel 4. 19 Cluster

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cluster 0** | **Cluster 1** | **Cluster 2** | **Cluster 3** | **Cluster 4** | **Cluster 5** |
| Titikpusat D2  D2 | Titik pusat D3  D0 | Titik pusat D5  D5 | Titik pusat D6  D4  D7 | Titik pusat D8  D8 | Titik pusat D9  D9 |

Tabel 4.20 Mencari anggota baru dalam cluster

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Jarak medoid D2 | Jarak medoid D3 | Jarak medoid D5 | Jarak medoid D6 | Jarak medoid D8 | Jarak medoid D9 | cluster |
| D0 | 0 | 0.211 | 0 | 0.095 | 0 | 0 | 1 |
| D1 | 0 | 0 | 0 | 0.270 | 0.135 | 0 | 3 |
| D2 | 1.000 | 0 | 0 | 0 | 0.107 | 0.125 | 0 |
| D3 | 0 | 1.000 | 0 | 0.101 | 0 | 0 | 1 |
| D4 | 0.086 | 0.081 | 0.099 | 0.366 | 0.073 | 0.086 | 3 |
| D5 | 0 | 0 | 1.000 | 0 | 0.246 | 0.289 | 2 |
| D6 | 0 | 0.101 | 0 | 1.000 | 0 | 0 | 3 |
| D7 | 0.144 | 0 | 0 | 0.369 | 0 | 0.144 | 3 |
| D8 | 0.107 | 0 | 0.246 | 0 | 1.000 | 0.107 | 4 |
| D9 | 0.125 | 0 | 0.289 | 0 | 0.107 | 1.000 | 5 |
| ∑ | 1.462 | 1.393 | 1.634 | 2.199 | 1.668 | 1.751 |  |

Dari perhitungan iterasi maka didapatkan hasil setiap *cluster* seperti pada Table 4.21:

Tabel 4. 21 judul

|  |  |
| --- | --- |
| **ID** | **Cluster 0** |
| D2 | Perancangan Sistem Test Komputerisasi dan Pendukung Keputusan Penerimaan Pegawai |
|  | **Cluster 1** |
| D0 | Penerapan antarmuka bahasa alami dalam pencarian informasi skripsi pada suatu program studi |
| D3 | Pengolahan bahasa alami pada agen cerdas alat bantu konsultasi zakat |
|  | **Cluster 2** |
| D5 | Pengembangan template media pembelajaran berbasis flash |
|  | **Cluster 3** |
| D1 | System reminder aktifitas akademik dosen |
| D4 | Perancangan dan implementasi system pengenalan jenis kulit hewan untuk kerajinan kulit berbasis sistem cerdas |
| D6 | Membangun system informasi(SIE) dengan menggunakan pendekatan system cerdas |
| D7 | Perancangan visualisasi informasi untuk system evaluasi guru |
|  | **Cluster 4** |
| D8 | Pengembangan sistem penilaian kerja dosen dengan aspek ikd dan skp 2014 berbasis online |
|  | **Cluster 5** |
| D9 | Perancangan animasi wayang pendidikan sebagai pengembangan media edukasi sekolah |

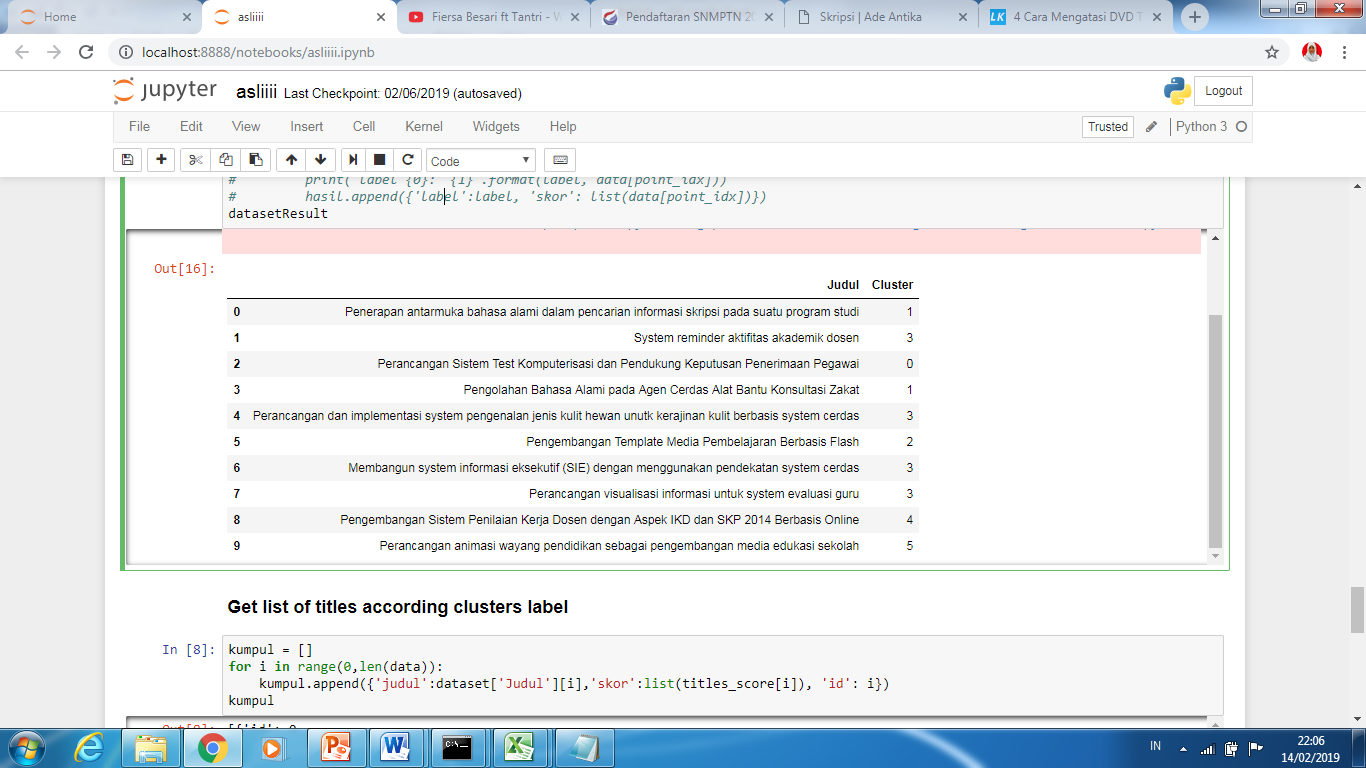
Setelah melakukan perhitungan hasil cosine yang ditunjukkan pada Tabel 4.2 *Cosine Similarity* , maka selanjutnya adalah mencari anggota cluster *K-Medoids* nilai yang diambil adalah hasil dari *Cosine.* Hasil cluster yang terbentuk ditunjukkan pada Tabel 4.21 judul. Dan proses pada sistem ditunjukkan pada Listing 4.4 *K-Medoids* dan 4.4 hasil *cluster*

|  |  |
| --- | --- |
| 1 | data = np.array(titles\_score) |
| 2 | datasetResult = dataset.copy() |
| 3 | datasetResult['Cluster'] = 0 |
| 4 | M, C = kMedoids(D, 6) |
| 5 | print("<Medoids:{}>".format(M)) |
| 6 | hasil = [] |
| 7 | for label in C: |
| 8 | for point\_idx in C[label]: |
| 9 | datasetResult["Cluster"][point\_idx] = label |
| 10 | datasetResult |

Listing 4.4 *K-Medoids*

Keterangan :

1. Baris 1, variabel yang menampung title\_score untuk nilai cosine
2. Baris 2- 3, dataResult untuk dataset menampung judul cluster
3. Baris 4, fungsi K-medoids dan inputan pada cluster
4. Baris 7- 10 , melakukan perulangan untuk dataset judul dan nama cluster.



Gambar 4.4 Hasil *cluster*

* 1. **Kebutuhan Software**

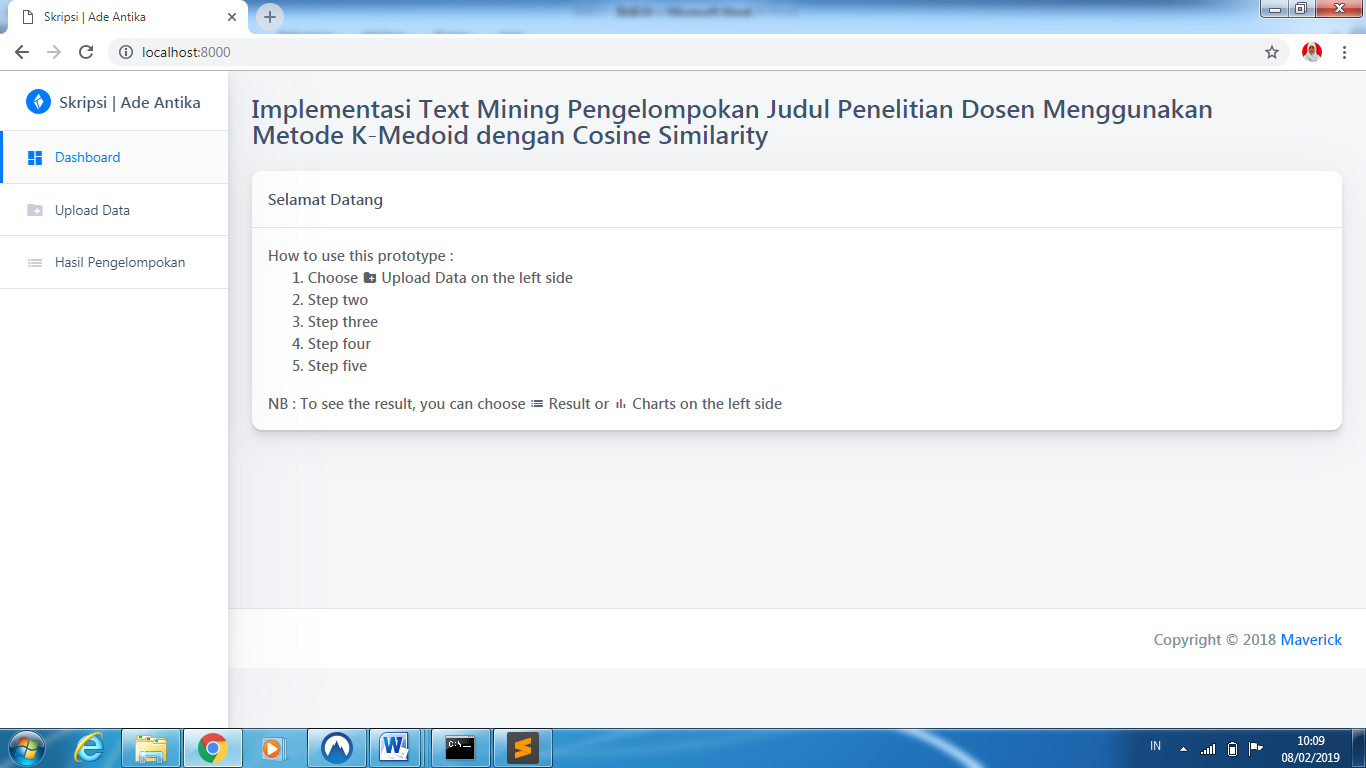
Dalam penelitian ini menggunakan *Micro web Framework* dengan bahasa pemrograman *Python. Library* yang yang digunakan sebagai berikut

1. *Library Pandas*
2. *Library Openpyxl*
3. *Library Scikitlearn*
4. *Library NLTK*
5. *Library Sastrawi*
6. *Library Numpy*
7. *Library K-Medoids* <https://github.com/letiantian/kmedoids/blob/master/kmedoids.py>
8. *Library Engine*
   1. **IMPLEMENTASI**

Implementasi Program “*Text Mining* menggunakan *Cosine Similarity* dengan metode *K-Medoids*” untuk pengelompokkan judul penelitian dosen ini menggunakan *Micro Web Framework Flask* dengan Bahasa pemrograman *Python.* Implementsi aplikasi *teks mining* berbasis web

* + 1. **Dashboard**

Implementasi untuk dashboard sistem terdiri dari menu upload data, Hasil pengelompokkan. Upload data digunakan untuk melakukan load data excel ke sistem. Menu hasil pengelompokkan digunakan untuk melihat hasil cluster *K-Medoids* yang terbentuk. Tampilan dapat dilihat pada Gambar 4.5 Interface Dasboard dan Listing 4.5 Listing home.

Gambar 4.5 Interface Dashboard

|  |  |
| --- | --- |
| 1 | @app.route('/', methods=['GET', 'POST']) |
| 2 | def home(): |
| 3 | return render\_template("index.html") |

Gambar 4.5 Listing Home

Keterangan:

1. Baris 2 - 3 membuat tampilanDashboard
   * 1. **Upload Data**

Upload data adalah halaman untuk mengupload data excel. Terdapat button upload untuk melakukan *action* Upload. Implementasi untuk upload data sistem dapat dilihat pada Gambar 4.6 Upload data dan 4.6 Listing upload.

Gambar 4.6 Upload Data

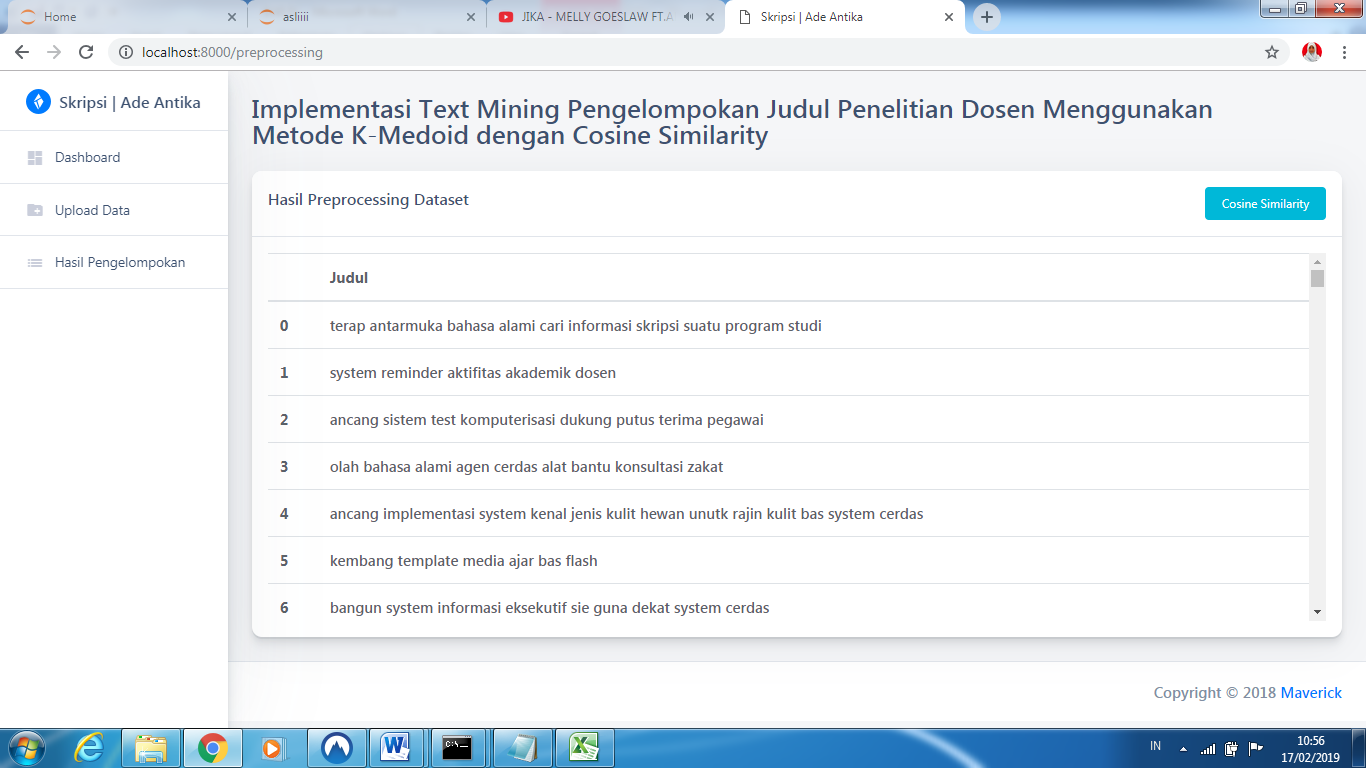
|  |  |
| --- | --- |
| 1 | @app.route('/upload', methods=['GET', 'POST']) |
| 2 | def upload(): |
| 3 | if request.method == 'POST': |
| 4 | global dataset |
| 5 | global dataset\_name |
| 6 | dataload = request.files['file'] |
| 7 | dataload.save(os.path.join('app/data', 'data.xlsx')) |
| 8 | wb = load\_workbook(filename='app/data/data.xlsx') |
| 9 | dataset = pd.DataFrame(wb['Sheet1'].values) |
| 10 | dataset.columns = ['Judul'] |
| 11 | count\_dataset = Dataset.query.count() |
| 12 | dataset\_name = "Dataset\_{}".format(count\_dataset + 1) |
| 13 | try: |
| 14 | data\_model = Dataset(name=dataset\_name) |
| 15 | db.session.add(data\_model) |
| 16 | db.session.commit() |
| 17 | except Exception as e: |
| 18 | print("Failed to add data | {}".format(e)) |
| 19 | print(dataset) |
| 20 | return render\_template("uploaded\_files.html",tables=[dataset.to\_html(classes="table mb-0 border-0 table-responsive", justify='unset').replace('border="1"', "")]) |
| 21 | else: |
| 22 | return render\_template("upload.html") |

Gambar 4.6 Listing upload

Keterangan:

1. Baris 3 – 6 menyimpan nama *file* kedalam variabel dataload
2. Baris 7 untuk menyimpan data upload ke dalam *data*
3. Baris 8 – 19 digunakan untuk melakukan import data dari *excel* dan disimpan dalam *variabel* wb kemudian variable sheet\_range yang akan menampung data dari *sheet* mana yang akan diambil dalam *file* *excel* dan melakukan *parsing* data terstruktur kedalam bentuk kolom dan baris dan akan di simpan kedalam *database*.
   * 1. **Preprocessing**

Setelah melakukan proses upload ke sistem, maka tahapan selanjutnya adalah melakukan tahapan Preprocessing untuk mengubah text yang tidak terstruktur menjadi terstruktur melalui proses tahapan tokenisasi, *filtering dan stemming*. Implementasi sistem untuk preprocessing dapat dilihat pada Gambar 4.7 Preprocessing dan gambar 4.7 Listing preprocessing

Gambar 4.7 Preprocessing

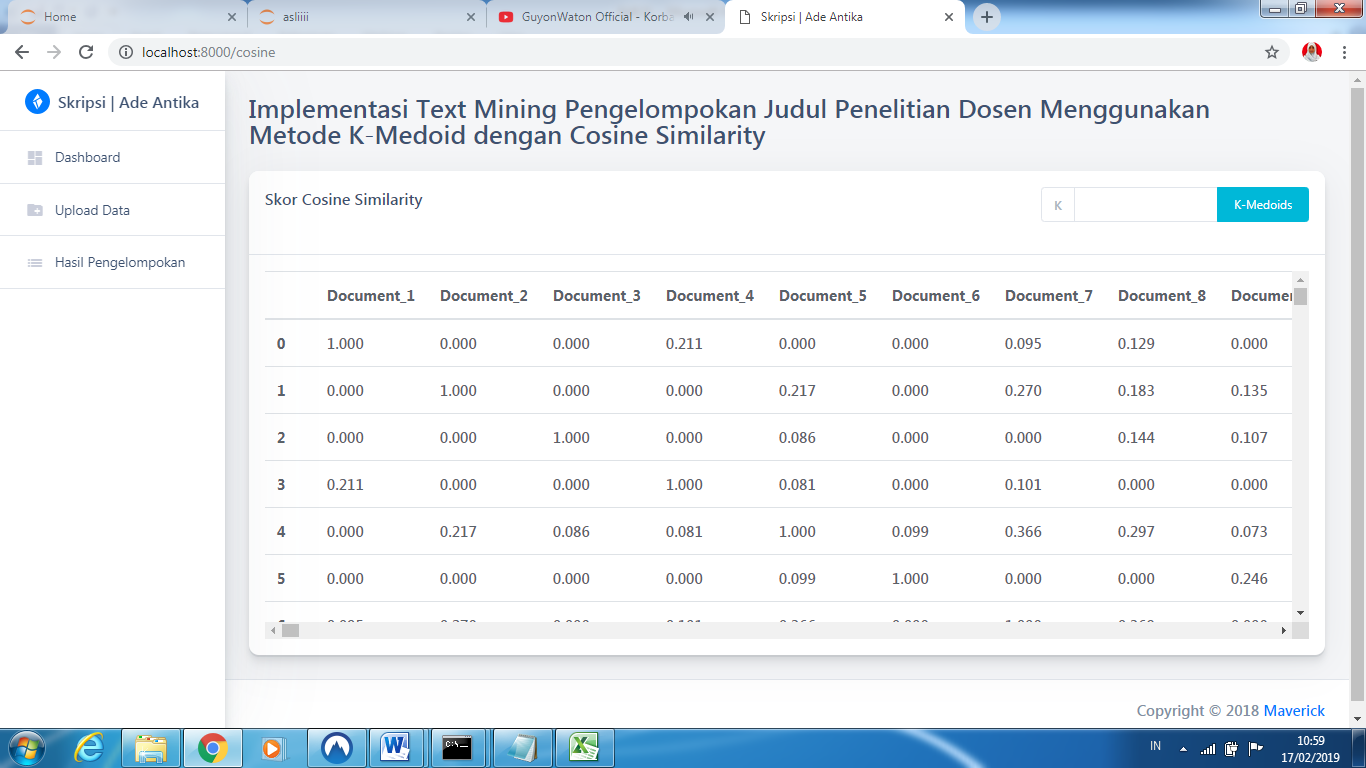
|  |  |
| --- | --- |
| 1 | @app.route('/preprocessing', methods=['GET', 'POST']) |
| 2 | def preprocessing(): |
| 3 | global dataset |
| 4 | global pre\_judul |
| 5 | list\_pre\_judul = [] |
| 6 | try: |
| 7 | for data in dataset["Judul"]: |
| 8 | list\_pre\_judul.append(preprocess(data)) |
| 9 | except Exception as e: |
| 10 | list\_pre\_judul = ["Data belum di upload"] |
| 11 | pre\_judul = pd.DataFrame(list\_pre\_judul) |
| 12 | pre\_judul.columns = ["Judul"] |
| 13 | print(dataset) |
| 14 | Returnrender\_template("preprocessing.html",tables=[pre\_judul.to\_html(classes="table mb-0 border-0 table-responsive", justify='unset').replace('border="1"', "")]) |

Gambar 4.7 Listing preprocessing

Keterangan:

1. Baris 3 - 5 menyimpan nama variabel dan mengisi list pre judul
2. Baris 7-8 proses preprocessing mengubah bentuk kata yang tidak terstruktur menjadi terstruktur
   * 1. ***Cosine Similarity***

*Cosine Similarity* adalah proses untuk mencari bobot setiap data bertujuan untuk mengecek kemiripan antara dokumen satu dengan yang lainnya berdasarkan kedekatan bobot dokumen. Implementasi *similarity* dapat dilihat pada Gambar4.8 dan *coding* pada *Listing* 4.8.

Gambar 4.8 *Cosine Similarity*

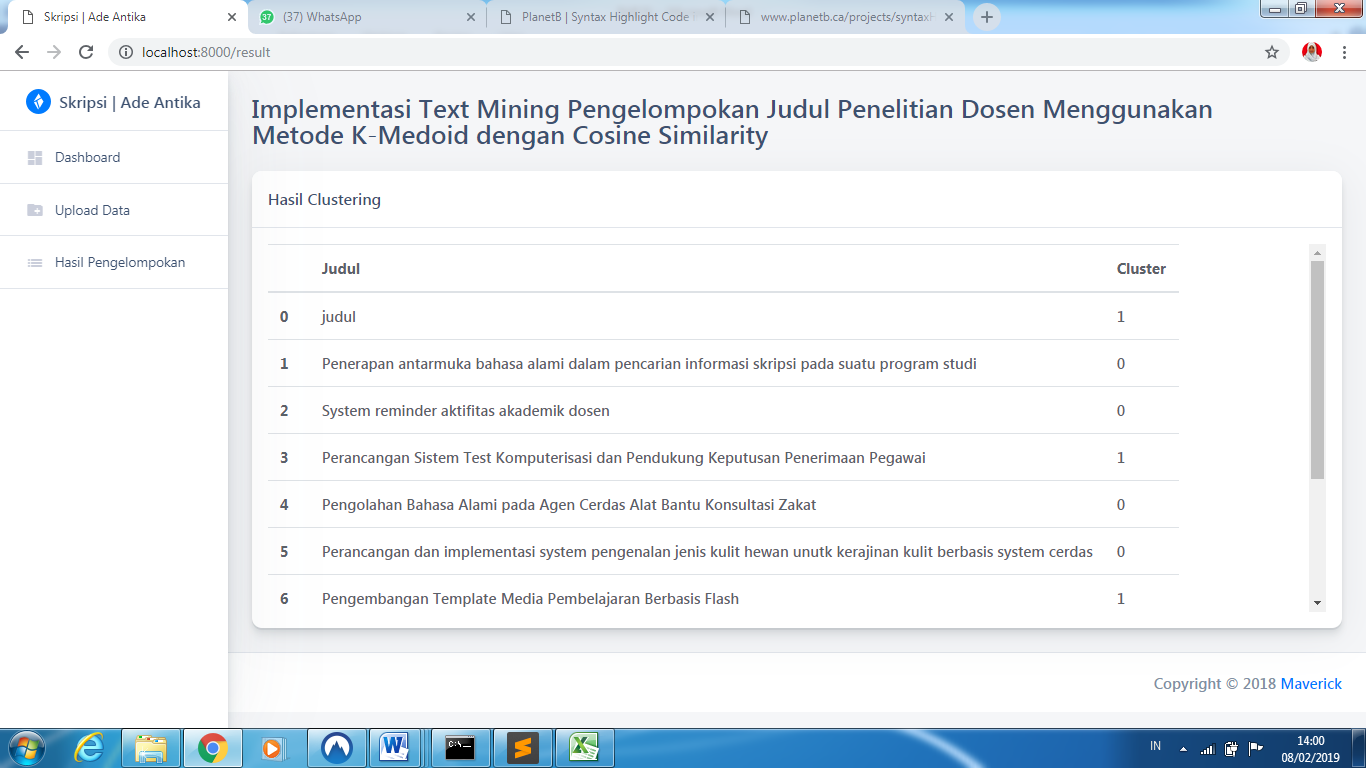
|  |  |
| --- | --- |
| 1 | @app.route('/cosine', methods=['GET', 'POST']) |
| 2 | def cosine(): |
| 3 | global pre\_judul |
| 4 | global titles\_score |
| 5 | # Call Engine from Custom Libraries |
| 6 | engine = Engine() |
| 7 | # Define data uji |
| 8 | list\_dokumen = [str(x) for x in pre\_judul['Judul']] |
| 9 | list\_datauji = [str(x) for x in pre\_judul['Judul']] |
| 10 | columnsName = [] |
| 11 | for i, doc in enumerate(list\_dokumen): |
| 12 | engine.addDocument(doc) |
| 13 | columnsName.append("Document\_{}".format(i + 1)) |
| 14 | for doc in list\_datauji: |
| 15 | engine.setQuery(doc) |
| 16 | titles\_score = engine.process\_score() |
| 17 | titlesScoreDf = pd.DataFrame(titles\_score) |
| 18 | titlesScoreDf.columns = columnsName |
| 19 | return render\_template('cosine.html', tables=[titlesScoreDf.to\_html(classes="table mb-0 border-0 table-responsive", justify='unset').replace('border="1"', "")]) |

Gambar listing 4.8 *cosine similarity*

Keterangan :

1. Baris 3 - 4 mengambil data pre\_judul dan titile\_score
2. Baris 6 - 9 proses list dokumen dan list data uji
3. Baris 11 – 16 proses menghitung nilai cosine pada setiap dokumen dan memasukkan nilai ke dalan tabel.
4. Baris 17 – 18 proses memberikan judul pada setiap kolom kata.
   * 1. **Clustering**

*Clustering* adalah proses untuk mengecek kemiripan bobot setiap dokumen dan menggabungkan dokumen yang dianggap bobotnya mirip menjadi satu *cluster*. Implementasi *clustering* pada gambar4.9 dan *listing* 4.9*.*

Gambar 4.9 *clustering*

|  |  |
| --- | --- |
| 1 | @app.route('/result', methods=['GET','POST']) |
| 2 | def result(): |
| 3 | if request.method == "POST": |
| 4 | global titles\_score |
| 5 | global dataset |
| 6 | k = int(request.form['k']) |
| 7 | data = np.array(titles\_score) |
| 8 | datasetResult = dataset.copy() |
| 9 | datasetResult['Cluster'] = 0 |
| 10 | D = pairwise\_distances(data, metric='cosine') |
| 11 | M, C = kMedoids(D, k) |
| 12 | data\_model = Dataset.query.filter\_by(name=dataset\_name).first() |
| 13 | try: |
| 14 | data\_model.medoid = str(M) |
| 15 | db.session.commit() |
| 16 | except Exception as e: |
| 17 | print("Failed to update medoid: {}".format(e)) |
| 18 | print("<Medoids:{}>".format(M)) |
| 19 | hasil = [] |
| 20 | for label in C: |
| 21 | for point\_idx in C[label]: |
| 22 | datasetResult["Cluster"][point\_idx] = label |
| 23 | count\_clusterdb = Cluster.query.count() |
| 24 | clusterdb\_name = "Clustering\_{}".format(count\_clusterdb + 1) |
| 25 | try: |
| 26 | clusterdb = Cluster(name=clusterdb\_name, number\_of\_cluster=len(C), dataset\_id=data\_model.id) |
| 27 | db.session.add(clusterdb) |
| 28 | db.session.commit() |
| 29 | except Exception as e: |
| 30 | print("Failed to add data | {}".format(e)) |
| 31 | clusterdb = Cluster.query.filter\_by(name=clusterdb\_name).first() |
| 32 | for i in range(len(datasetResult["Cluster"])): |
| 33 | cluster\_data = ClusterData(title=datasetResult["Judul"][i], cluster=int(datasetResult["Cluster"][i]), cluster\_id=clusterdb.id) |
| 34 | try: |
| 35 | db.session.add(cluster\_data) |
| 36 | db.session.commit() |
| 37 | except Exception as e: |
| 38 | print("Failed to add data | {}".format(e) print(datasetResult) |
| 39 | return render\_template('result.html', tables=[datasetResult.to\_html(classes="table mb-0 border-0 table-responsive", justify='unset').replace('border="1"', "")]) |

Listing 4.9 *K-Medoids*

Keterangan :

1. Baris 3 - 9 menyimpan ke sebuah *variable* hasil inputan *user*
2. Baris 11 *function kmedoids* mengambil dari library untuk proses mengelompokkan data
3. Baris 13 – 20 proses memecah kelompok sesuai jumlah kelompok yang telah diinputkan
4. Baris 10 - 15 mengambil dari database data awal
5. Baris 22 – 34 menampilkan hasil pengelompokkan ke dalam dataframe
6. Baris 35- 38 memberikan label pada kelompok yang terbentuk
   1. **PENGUJIAN**

Pengujian dilakukan menggunakan *purity test* untuk mengetahui akurasi dari hasil yang di tampilkan oleh system dan pengamatan langsung pada data. Pengujian dilakukan untuk menguji baik atau tidaknya metode yang digunakan dalam pengelompokan judul penelitian dosen. Pada perhitungan *purity test* akan menghasilkan *range* nilai dari 0 hingga 1, 0 diartikan dengan nilai akurasi tidak bagus sedangkan semakin mendekati nilai 1 maka dapat disimpulkan pengujian akurat atau pengelompokan data

Dimana data yang diuji sebanyak 517 data, yaitu data judul penelitian dosen pada tahun 2012 hingga tahun 2017. Pengelompokkan dibagi menjadi 6 cluster yang mengacu pada bidang penelitian yaitu Teknologi informasi, sosial dan pendidikan, ekonomi, agama, kesehatan dan obat, dan hukum. Hasil dari *purity test* dapat dilihat pada table 4.23.

Tabel 4. 23 tabel pengujian

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Bidang | Asli | System | kesesuaian |
|  | Teknologi informasi | 65 | 78 | 56 |
|  | Sosial dan pendidikan | 230 | 217 | 207 |
|  | Ekonomi | 78 | 80 | 75 |
|  | Agama | 28 | 28 | 24 |
|  | Kesehatan dan obat | 74 | 74 | 73 |
|  | Hukum | 42 | 40 | 37 |
|  | ∑ | 517 | 517 | 472 |

Dimana :

{w1,w2, ...wk} adalah kumpulan *cluster*

{c1,c2,... cj} adalah kumpulan kelas acuan

semua data yang di uji

adalah kumpulan objek pada *cluster* Wk

adalah kumpulan objek pada kumpulan kelas acuan

Sesuai dengan hasil *cluster* diatas hasilnya sebagai berikut :

Pengelompokan data asli dilakukan secara manual agar dapat dibandingkan dengan hasil dari pengelompokan system menggunakan *cosine similarity* dengan metode *K-Medoids* dan didapatkan hasil *purity test* dari pengujian yaitu .

* 1. **EVALUASI POLA**

Pada tabel 4.20 hasil dari *purity test* pola yang terbentuk adalah sebagai berikut:

1. Pola yang terbentuk dari pengelompokan data sebanyak 517 data dibagi menjadi 6 *cluster* hasilnya data belum terpisah seluruhnya menjadi kelompok bidang penelitian.
2. Dilakukan beberapa percobaan menggunakan 517 data dengan jumlah *cluster* 6, hasil perobaan *test* *purity* menunjukkan nilai akurasi sebesar Pengelompokan yang dihasilkan dari akurasi nilai tertinggi yaitu:
3. *Cluster* 0 dengan jumlah data 15, teknologi informasi yang menjadi dominan dan sisanya data yang meliputi bidang lain.
4. *Cluster 1* dengan jumlah data 108, sosial dan pendidikan yang menjadi dominan, sisanya data yang meliputi bidang lain.
5. *Cluster 2* dengan jumlah data 5, Ekonomi yang menjadi dominan .
6. *Cluster 3* dengan jumlah data 86, *cluster 3* menjadi cluster dengan label ekonomi namun dominan data pada *cluster* 3 lebih condong bidang pendidikan dikarenakan data bidang sosial dan pendidikan cenderung mengelompok pada *cluster* 1.
7. *Cluster 4* dengan jumlah data 69 sebanyak 53 data pendidikan dan sosial sisanya data yang meliputi bidang lain.
8. *Cluster 5* dengan jumlah data 227, sebanyak 58 data pendidikan dan sosial , 44 data kesehatan dan obat, sisanya meliputi bidang teknologi dan informasi.