ANALISIS DATA UJARAN KEBENCIAN

Makalah Ini Disusun untuk Memenuhi UAS Mata Kuliah

Metode Kecerdasan Buatan

Dosen Pengampu : Muhammad Irvan Septiar Musti, S.Si, M.Si

Di susun oleh :

Fida Suci Rahmani	11190940000027
Rosa Amalia Nursinta	11190940000041
Elviana Saputri	11190940000043
Ghina Rahmah	11190940000053
Meissy Astariva Putri	11190940000063

PROGRAM STUDI MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH JAKARTA

2022

KATA PENGANTAR

Assalammu'alaikum Warahmatullahi Wabarakatuh

Puji syukur mari kita panjatkan kepada Allah SWT yang Maha Esa karena atas berkat rahmat dan karunia-Nya kami dapat menyelesaikan penyusunan makalah dengan judul "Analisis Data Ujaran Kebencian" dapat dikerjakan dengan tepat waktu dan dengan sebaik-baiknya.

Makalah ini merupakan tugas yang harus diselesaikan dalam mata kuliah Metode Kecerdasan Buatan dengan dosen pengampunya adalah Bapak Muhammad Irvan Septiar Musti, S.Si, M.Si. Harapan kami tentunya berharap agar makalah ini dapat membantu dan menambah wawasan bagi para pembaca.

Kami menyadari penuh bahwa dalam penyusunan project ini masih terdapat banyak kekurangan. Kami mengharapkan kritik dan saran yang membangun dari pembaca agar kedepannya kami bisa melakukan perbaikan untuk mendapatkan hasil yang lebih baik. Terakhir, kami berharap semoga laporan ini dapat bermanfaat dan dapat tercapai sesuai dengan yang diharapkan.

Wassalammu'alaikum Warahmatullahi Wabarakatuh

Tangerang, 28 Juni 2022

Penulis

PEMBAHASAN

A. Sample Dataset

	created_at	date	time	user_id	username	name	text	mentions	urls	photos	...	quote_url	near	geo	source	user_rt_id	user_rt	retweet_id	reply_to	retweet_date	label
0	2021-06-13 22:36:12 SE Asia Standard Time	6/13/2021	22:36:12	3.5E+07	pasadenasl	Mith	@Anonymous_2024 Pemerintah harus revisi ulang ...	[]	[]	[]	...	None	NaN	NaN	NaN	NaN	NaN	NaN	[{'screen_name': 'Anonymous_2024', 'name': 'An...	NaN	N
1	2021-06-13 21:52:52 SE Asia Standard Time	6/13/2021	21:52:52	1.4E+18	stddevx	St. Devyz	@molatv_living djarum cina bangsat gak becus, ...	[]	[]	[]	...	None	NaN	NaN	NaN	NaN	NaN	NaN	[]	NaN	H
2	2021-06-13 21:48:15 SE Asia Standard Time	6/13/2021	21:48:15	7.9E+17	beige_chocolate	Cheesecakebiss	@opiedupdidup @diyaaa_nair @JanaaniBala Yang b...	[]	[]	[]	...	None	NaN	NaN	NaN	NaN	NaN	NaN	[{'screen_name': 'opiedupdidup', 'name': 'Tmay...	NaN	H
3	2021-06-13 21:31:31 SE Asia Standard Time	6/13/2021	21:31:31	3E+09	_hidayahazhar	Ù‡ÙŠØ¯ Ø§ÙŠÙ‡ Ø§Ø²Ù‡ Ø±	@AmeirHasif Sbb takde lg yg berani bawa isu ke...	[]	[]	[]	...	None	NaN	NaN	NaN	NaN	NaN	NaN	[{'screen_name': 'AmeirHasif', 'name': 'AmeiR'...	NaN	N

B. Pembahasan Soal

1. Analisis Data Exploratif

Berikut kami tampilkan dataset yang diperoleh dari dataLabeled.Json Dataset ini berjumlah 26 variabel yaitu : created_at, date, time, user_id, username, name, text, mentions, urls, photos, replies_count, replies_count, retweets_count, likes_count, hashtags, link ,retweet, near, quote_url, geo, source, user_rt_id, user_rt , retweet_id ,reply_to retweet_date, label.

1.1. Mendapatkan Tweet dengan nilai retweet tertinggi

```
retweet = data[['text', 'retweets_count', 'label']]
```

```
retweet.sort_values(by=['retweets_count'], ascending=False).head()
```

	text	retweets_count	label
6995	HAI MALING2 UANG RAKYAT HAI YG NILEP DUIT RAKY...	2005	H
9764	Msh kaget baca org yg mempertanyakan apakah su...	1867	N
8300	Untuk saudara-saudaraku umat Buddha, Tri Suci ...	1280	N
7372	Dis! gua berharap bngt buat kedepannya generas...	1034	N
9580	Semoga jodohmu termasuk seseorang yang rajin d...	892	N

1.2. Mendapatkan Tweet dengan nilai likes tertinggi

```
likes = data[['text', 'likes_count', 'label']]
```

```
likes.sort_values(by=['likes_count'], ascending=False).head()
```

	text	likes_count	label
8300	Untuk saudara-saudaraku umat Buddha, Tri Suci ...	7986	N
9764	Msh kaget baca org yg mempertanyakan apakah su...	6209	N
9580	Semoga jodohmu termasuk seseorang yang rajin d...	4967	N
7372	Dis! gua berharap bngt buat kedepannya generas...	3881	N
6995	HAI MALING2 UANG RAKYAT HAI YG NILEP DUIT RAKY...	3350	H

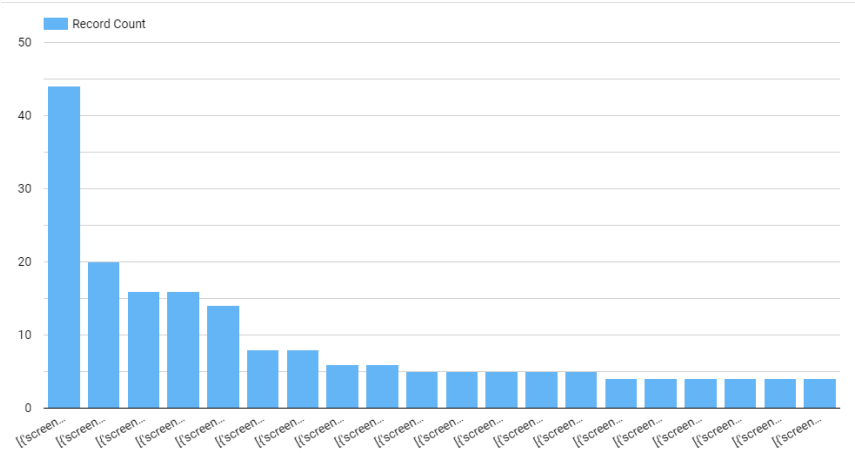
1.3. Mendapatkan Tweet dengan nilai reply tertinggi

```
replies = data[['text', 'replies_count', 'label']]
```

```
replies.sort_values(by=['replies_count'], ascending=False).head()
```

	text	replies_count	label
10805	Pernyataan Hendropriyono itu menunjukkan minim...	693	N
8300	Untuk saudara-saudaraku umat Buddha, Tri Suci ...	398	N
6995	HAI MALING2 UANG RAKYAT HAI YG NILEP DUIT RAKY...	354	H
9580	Semoga jodohmu termasuk seseorang yang rajin d...	317	N
9764	Msh kaget baca org yg mempertanyakan apakah su...	239	N

1.4. Mendapatkan username yang paling sering di mention di dalam konten tweet

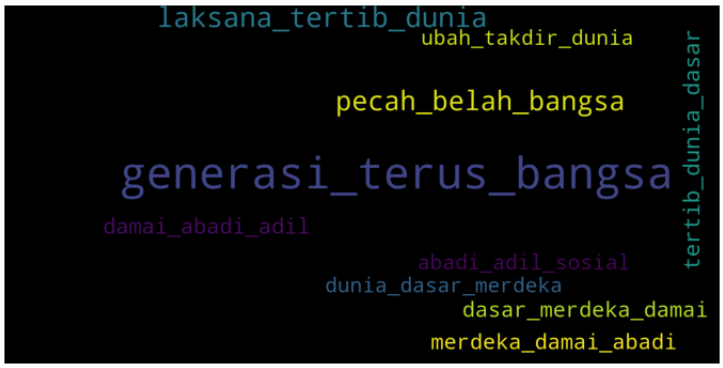


Akun yang paling sering di *mention* adalah akun ‘Jokowi’ sebanyak 44 akun, diikuti oleh ‘Denny Siregar’ sebanyak 20 akun dan selanjutnya ada beberapa orang dengan jumlah ter*mention* yang sama yaitu sebanyak 16 akun. Diagram di atas hanya menampilkan 20 akun dengan *mention* terbanyak.

1.5. Menampilkan Top 10 kata menggunakan unigram, bigram dan trigam dalam bentuk wordcloud dan bar chart.

Biagram

Unigram



Trigram

1.6. Mendapatkan analisis deret waktu berkaitan dengan peak time (tanggal dengan konten hatespeech terbanyak)

```
]: hate.describe()
```

]:

	user_id	replies_count	retweets_count	likes_count	near	geo	source	user_rt_id	user_rt	retweet_id	retweet_date
count	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1.7. Mendapatkan analisis deret waktu berkaitan dengan peak time (tanggal dengan konten Non-hatespeech terbanyak)

	user_id	replies_count	retweets_count	likes_count	near	geo	source	user_rt_id	user_rt	retweet_id	retweet_date
count	6.405000e+03	6405.000000	6405.000000	6405.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mean	7.241404e+17	0.983763	2.570336	11.513973	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	5.943485e+17	12.273713	38.193992	170.904549	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	6.763992e+06	0.000000	0.000000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	1.354197e+09	0.000000	0.000000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	1.010744e+18	0.000000	0.000000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	1.268521e+18	1.000000	0.000000	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	1.404042e+18	693.000000	1867.000000	7986.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2. Vektorisasi (Feature Extraction)

2.1. TF-IDF (Term Frequency-Inverse Document Frequency)

Teori Term Frequency (TF) merupakan pembobotan frekuensi dari munculnya setiap kata dalam sebuah dokumen. Semakin sering munculnya suatu kata (TF tinggi) pada dokumen tersebut, maka bobotnya semakin besar pula sehingga memberikan nilai kesesuaian yang tinggi.

Terdapat beberapa formula Term Frequensy (TF) :

1. TF biner (binary TF)

   TF biner memfokuskan jika suatu kata ada pada dokumen tersebut maka akan diberi nilai (1), jika tidak maka nilainya (0)
2. TF murni (raw TF)

   Nilai TF disesuaikan dengan jumlah kemunculan kata pada dokumen tersebut. Misalnya, muncul sebanyak (3) kali maka kata tersebut bernilai (3).
3. TF logaritmik

   Tf logaritmik digunakan dalam dokumen yang mengandung sedikit kata dalam query dengan frekuensi tinggi sehingga terhindar dari dominansi dokumen.
4. Nilai **ft,d** yaitu frekuensi term atau kata (t) dalam document (d). Jika suatu kata muncul dalam sebuah dokumen sebanyak 3 kali maka bobotnya = 1 + log (3) = 1,477 dan sebaliknya jika suatu kata tidak muncul maka bobotnya nol (0).
5. TF normalisasi

Tf normalisasi memakai rasio antara frekuensi sebuah kata dan nilai maksimum dari semua kumpulan frekuensi term dalam sebuah dokumen tersebut.

Term Frequency-Inverse Document Frequency yaitu ukuran statistik yang menggambarkan pentingnya istilah untuk dokumen dalam koleksi atau korpus. Parameter ini digunakan dalam faktor pembobot pencarian informasi, penambangan teks dan pemodelan pengguna. Nilai tf-idf bertambah sejalan dengan muncul banyaknya istilah yang bergantung pada jumlah dokumen dalam korpus.

Istilah Frekuensi

$tf(t,d)$ adalah frekuensi istilah dari t

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$f_{t,d}$ yaitu pencacahan awal istilah pada dokumen dimana banyaknya istilah t pada dokumen d. Semakin tinggi istilah, maka semakin besar nilai tf-nya. Berikut skema dan bobot pada frekuensi istilah :

Skema	Bobot tf
Biner	0,1
Pencacahan mentah	$f_{t,d}$
Frekuensi istilah	$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
Penormalan log	$\log(1 + f_{t,d})$
Penormalan ganda 0,5	$0{,}5 + 0{,}5x\frac{f_{t,d}}{\max\{t' \in d\} f_{t',d}}$

Penormalan ganda K	$K + (1 - K)\frac{f_{t,d}}{\max\{t' \in d\} f_{t',d}}$

Inversi Frekuensi Dokumen

Inversi Frekuensi Dokumen disimbolkan dengan $idf(t, D)$ dimana besaran informasi oleh istilah t yaitu berapa banyak munculnya pada dokumen. Semakin sedikit keluarnya istilah pada dokumen maka semakin tinggi nilai idf-nya. Nilai logaritma dari frekuensi dokumen adalah sebagai berikut :

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

Keterangan :

$\{d \epsilon D : t \epsilon d\}$: Himpunan dokumen d pada D yang mempunyai istilah t

Skema	Bobot idf
Basis satu	1
Inversi frekuensi rendah	$log\frac{N}{n_t} = -log\frac{n_t}{N}$
Halus	$log\left(\frac{N}{1 + n_t}\right) + 1$
Maks	$log\left(\frac{\max\{t' \in d\} n_{t'}}{1 + n_t}\right)$
Probabilistik	$log\frac{N - n_t}{n_t}$

Keterangan $n_t = \{d \epsilon D : t \epsilon d\}$

Frekuensi Istilah-inversi frekuensi dokumen (Tf-idf)

Rumus Tf-idf sebagai berikut :

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

Nilai Tf-idf akan besar ketika istilah sering muncul (tf tinggi), akan tetapi dalam dokumen yang sedikit (idf tinggi atai df rendah).

Skema Tf-idf sebagai berikut :

skema	Bobot istilah dokumen (d)	Bobot istilah query (q)
1	$f_{t,d}.\ log\frac{n_t}{N}$	$\left(0{,}5 + 0{,}5x\frac{f_{t,d}}{max_t f_{t,q}}\right) x\ log\frac{N}{n_t}$
2	$\log\left(1 + f_{t,d}\right)$	$log\left(1 + \frac{N}{n_t}\right)$
3	$\left(1 + \log f_{t,d}\right) X\ log\frac{N}{n_t}$	$\left(1 + \log f_{t,q}\right) X\ log\frac{N}{n_t}$

2.2 Perhitungan Manual Tf-idf

Dokumen 1

Istilah	Jumlah
Saya	1
Sedang	1
Belajar	2
Menghitung	1

Dokumen 2

Istilah	Jumlah
Saya	1
Sedang	1
Menbaca	3
Kalimat	2

Langkah-langkah menghitung tf-idf :

Didalam frekuensi mentahnya, Tf yaitu frekuensi istilah *saya* dalam tiap dokumen . Pada dokumen 1 dan dokumen 2 istilah *saya* muncul sekali. Pada dokumen 2 mempunyai kata yang lebih banyak daripada dokumen 1, maka dokumen 2 memiliki frekuensi relatife lebih kecil.

$$tf(\text{"saya"}, d_1) = \frac{1}{5} = 0{,}2$$

$$tf(\text{"saya"}, d_2) = \frac{1}{7} = 0{,}14$$

Setiap korpus nilai idf akan tetap dan mengikuti jumlah dokumen yang memiliki istilah *saya*. Korpus dimana semua dokumennya mempunyai istilah *saya*.

$$idf(\text{"saya"}, D) = log\left(\frac{2}{2}\right) = 0$$

Jadi, diperoleh nilai tf-idf dengan istilah *saya* yaitu nol (0). Istilah "*saya*" ini tidak begitu penting karena muncul di semua dokumen.

$$tfidf(\text{"saya"}, d_1, D) = 0{,}2 \times 0 = 0$$

$$tfidf(\text{"saya"}, d_2, D) = 0{,}14 \times 0 = 0$$

Diberikan contoh lainnya yaitu istilah *membaca* muncul sebanyak tiga kali yang hanya terdapat pada dokumen 2.

$$tf(\text{"membaca"}, d_1) = \frac{0}{5} = 0$$

$$tf(\text{"membaca"}, d_2) = \frac{3}{7} \approx 0{,}429$$

$$idf(\text{"membaca"}, D) = log\left(\frac{2}{1}\right) = 0{,}301$$

Kemudian untuk mencari tf-idf yaitu :

$$tfidf(\text{"saya"}, d_1, D) = tf(\text{"membaca"}, d_1) \times idf(\text{"membaca"}, D) = 0 \times 0{,}301 = 0$$

$$tfidf(\text{"saya"}, d_2, D) = tf(\text{"membaca"}, d_2) \times idf(\text{"membaca"}, D) = 0{,}429 \times 0{,}301 \approx 0{,}129$$

2.3 Vektorisasi TF-IDF menggunakan Python

▾ Vektorisasi TF

```
[ ] # Membentuk verktorisasi IF

    tf = CountVectorizer(
        min_df = 5,
        max_df = 0.95,
        max_features = 8000,
        # stop_words = 'english'
    )
    tf.fit(data.text)
    tf_text = tf.transform(data.text)
```

▾ VEKTORISASI TF-IDF

```
[ ] # Membentuk verktorisasi TF-IDF

    tfidf = TfidfVectorizer(
        min_df = 5,
        max_df = 0.95,
        max_features = 8000,
        # stop_words = 'english'
    )
    tfidf.fit(data.text)
    tfidf_text = tfidf.transform(data.text)
```

2.4 Parameter min_df dan max_df

Parameter max_df untuk menghapus istilah yang muncul terlalu sering, juga dikenal sebagai "kata-kata berhenti khusus-corpus". Sebagai contoh:

max_df = 0.95 berarti "abaikan istilah yang muncul dalam lebih dari 95% dokumen".

Default untuk min_df dan max_df masing-masing 1 dan 1.0. Ini pada dasarnya mengatakan "Jika istilah saya ditemukan hanya dalam 1 dokumen, maka itu diabaikan. Demikian pula jika ditemukan di semua dokumen (100% atau 1,0) maka

diabaikan." min_df digunakan untuk menghapus istilah yang muncul terlalu jarang. Sebagai contoh:

min_df = 5 berarti "abaikan istilah yang muncul dalam kurang dari 5 dokumen".

3. Topic Detection – Minibatch K-Means

3.1. Latar Belakang

YANG PERTAMA ADA LATAR BELAKANG KLASTERISASI MINI BATCH K-MEANS

Klasfikasi merupakan pengelompokkan data berdasarkan kesamaan label, contohnya dalam mengklasifikasikan bunga iris, dengan mengelompokkan data yang telah berlabel yaitu pada bunga iris virginiga, versicolor, dan setosa. Sedangkan klasterisasi merupakan pengelompokkan data yang berdasarkan kemiripan data, bisa saja tidak ada labelnya. Kemiripan ini dilihat pada nilai atribut-atributnya (nilai pada kolom-kolomnya).

Kemudian bagaimana menentukan kemiripan data? . Pengukuran jarak data yang dihitung dengan persamaan Euclidean Distance

$$d = \sqrt{\sum_{N} (x_i - y_i)^2}$$

Pengukuran jarak dengan menjumlahkan selisih antara x_i atribut data yang pertama dan y_i atribut data yang kedua. Dimana i meyatakan index dari atributnya sampai atribut ke-n.

Algoritma pengelompokan Mini batch K-means adalah algoritma K-means yang dapat digunakan saat mengelompokkan pada himpunan data besar, mini batch k-means menggunakan batch data kecil, acak, ukuran tetap untuk disimpan dalam

memori, dan kemudian dengan setiap iterasi, sampel acak dari data dikumpulkan dan digunakan untuk memperbarui cluster. Terkadang kinerjanya lebih baik daripada algoritma K-means saat bekerja pada himpunan data besar karena tidak memerlukan pengulangan di seluruh himpunan data.

NEXT KE PERHITUNGAN MANUAL

Keuntungan utama menggunakan algoritma Mini-batch K-means adalah mengurangi biaya komputasi untuk menemukan cluster. Ide utama dari mini batch K-means adalah menggunakan batch kecil acak dari kumpulan data dengan ukuran tetap sehingga dapat disimpan dalam memori. Dalam setiap iterasi, sampel acak baru yg diperoleh dari kumpulan data digunakan untuk memperbarui centroid dan ini diulangi hingga konvergen. Setiap mini batch memperbarui centroid menggunakan metode penurunan gradien yang menerapkan learning rate untuk mendapatkan Konvergensi yg lebih cepat.

3.2. Pseudocode

UNTUK PSEUDOCODE NYA DAPA DILIHAT PADA GAMBAR BERIKUT

Input : data set $D = \{x_1, x_2, \dots, x_n\}$; jumlah klaster k; ukuran batch b; iterasi t

Output : klaster $C = \{c_1, c_2, \dots, c_k\}$

Process :

1: inisialisasi k centroid klaster $\mu = \{\mu_{c_1}, \mu_{c_2}, \dots, \mu_{c_k}\}$ dengan sampel k yang dipilih secara acak dari dataset D

2: $C_i \leftarrow \emptyset$ $(1 \leq i \leq k)$ # inisialisasi klaster

3: $N_{c_i} \leftarrow 0$ $(1 \leq i \leq k)$ # inisialisasi jumlah sampel untuk setiap klaster

4: **for** $j = 1,2, \dots, t$ **do**

5: $M \leftarrow \{x_m | 1 \leq m \leq b\}$ # M adalah batch dataset, dan x_m adalah sampel yang dipilih secara acak dari dataset D

6: **for** $m = 1,2, \dots, b$ **do** # step 6 sampai step 8 adalah untuk mendapatkan centroid pada setiap sampel pada batch set

7: $\mu_{c_i}(x_m) \leftarrow \frac{1}{|c_i|} \sum_{x_m \in c_i} x_m \, (x_m \in M)$ # $\mu(x_m)$ adalah centroid terdekat yang didapat terhadap data sampel x_m

8: **end for**

9: **for** $m = 1,2, \dots, b$ **do** # step 9 sampai step 14 adalah untuk memperbarui centroid pada setiap batch set

10: $\mu_{c_i} \leftarrow \mu_{c_i}(x_m)$ # mendapatkan centroid untuk sampel x_m

11: $N_{c_i} \leftarrow N_{c_i} + 1$ # memperbarui jumlah sampel pada setiap centroid

12: $\eta \leftarrow 1/N_{c_i}$ # menghitung learning rate untuk tiap centroid

13: $\mu_{c_i} \leftarrow (1 - \eta)\mu_{c_i} + \eta x_m$ # mengambil langkah gradien untuk memperbarui centroid

14: **end for**

15: **end for**

3.3. Contoh Perhitungan Manual Sederhana

Terdapat data awal sebanyak 20 data, dengan 2 buah features yaitu Indeks Kedalaman Kemiskinan dan Indeks Keparahan Kemiskinan.

Langkah 1 pada algoritma Mini Batch K-Means adalah membagi data ke dalam beberapa batch, pada contoh ini sebanyak 20 data dibagi kedalam 4 batch dengan ukuran setiap batch berisikan 5 data kemudian 20 data ini akan dikelompokkan ke dalam 2 klaster.

NEXT

	Data ke-	Indeks kedalaman Kemiskinan (x)	Indeks Keparahan Kemiskinan (y)
batch-1	1	1	1
	2	4	1
	3	6	1
	4	1	2
	5	2	3
batch-2	6	5	3
	7	2	5
	8	3	5
	9	2	6
	10	3	8
batch-3	11	2	3
	12	1	1
	13	4	1
	14	6	1
	15	2	5
batch-4	16	3	5
	17	3	8
	18	2	6
	19	6	1
	20	1	2

hal pertama: tentukan jumlah klaster
Jumlah klaster : K=2

Langkah 2 yaitu menentukan centroid awal untuk iterasi pertama. Pada contoh ini centroid batch pertama dipilih secara acak yaitu memisalkan data ke-2 dan data ke-4 sebagai centroid awal.

Langkah 3 yaitu menghitung jarak data terhadap centroid-centroid yang dipilih, pada contoh ini perhitungan jarak menggunakan *Euclidean distance*.

Langkah 4 yaitu membandingkan selisih jarak data terhadap centroid 1 dan centroid 2. Untuk menentukan kelompok mana yang akan menjadi klaster objek, pilih selisih jarak yang terkecil atau terdekat antara objek dan centroid. Selisih jarak terkecil terhadap centroid ke-n, akan menjadikan klaster n menjadi klaster objek tersebut.

batch-1
kemudian misalkan centoid pada batch-1 adalah data ke-2 , dan 4

	Data ke-	Indeks kedalaman Kemiskinan (x)	Indeks Keparahan Kemiskinan (y)
batch-1	1	1	1
	2	4	1
	3	6	1
	4	1	2
	5	2	3

Iterasi 1 Batch 1

Data ke-	centorid	x	y
2	c-1	4	1
4	c-2	1	2

Data ke-	Indeks kedalaman Kemiskinan (x)	Indeks Keparahan Kemiskinan (y)	c-1	c-2	Minimum	cluster
1	1	1	3	1	1	2
2	4	1	0	3,162278	0	1
3	6	1	2	5,09902	2	1
4	1	2	3,16227766	0	0	2
5	2	3	2,828427125	1,414214	1,414214	2

Data ke-	c-1	c-2
1		1
2	1	
3	1	
4		1
5		1

Langkah 5 melakukan iterasi kedua untuk mengetahui apakah klaster sudah konvergen atau belum. Iterasi kedua dilakukan dengan cara yang sama seperti iterasi pertama namun dengan titik centroid yang baru yaitu rata-rata dari setiap data di dalam klaster yang didapat pada iterasi pertama.

Langkah 6 jika pada iterasi kedua klaster yang didapat belum konvergen, lakukan pengulangan atau iterasi berikutnya sampai menemukan klaster yang tetap atau konvergen,

Iterasi 2 batch 1

Penentuan cluster baru	x	y
centorid 1 baru	5	1
centorid 2 baru	1,333333333	2

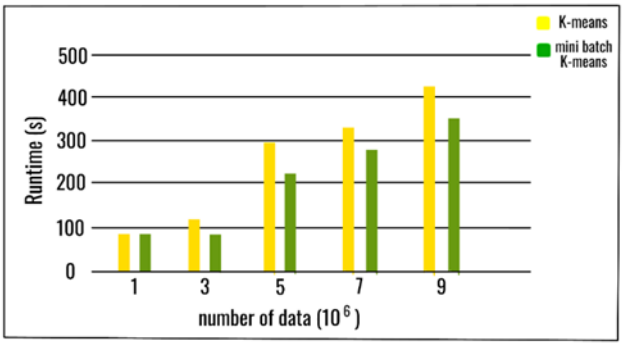
Data ke-	Indeks kedalaman Kemiskinan (x)	Indeks Keparahan Kemiskinan (y)	cluster awal	c-1	c-2	Minimum	cluster baru	ket
1	1	1	2	4	1,054093	1,054093	2	Tetap
2	4	1	1	1	2,848001	1	1	Tetap
3	6	1	1	1	4,772607	1	1	Tetap
4	1	2	2	4,123106	0,333333	0,333333	2	Tetap
5	2	3	2	3,605551	1,20185	1,20185	2	Tetap

Langkah 7 yaitu lakukan langkah yang sama pada batch-batch berikutnya seperti proses pada batch pertama. Klasterisasi berhenti jika seluruh data telah mendapat klaster yang tetap atau kovergen.

NEXT KE PSEUDOCODE

NEXT KE KELEBIHAN KEKURANGAN

3.4. Kelebihan dan Kekurangan



kekurangan : jumlah data (ukuran/size) pada tiap batchnya tidak boleh lebih kecil dari banyaknya klaster.

Kelebihan: Keuntungan utama menggunakan algoritma Mini-batch K-means adalah mengurangi biaya komputasi untuk menemukan cluster dan Terkadang kinerjanya lebih baik daripada algoritma K-mean saat bekerja pada himpunan data besar karena tidak memerlukan pengulangan di seluruh himpunan data sehingga konvergen lebih cepat.

NEXT

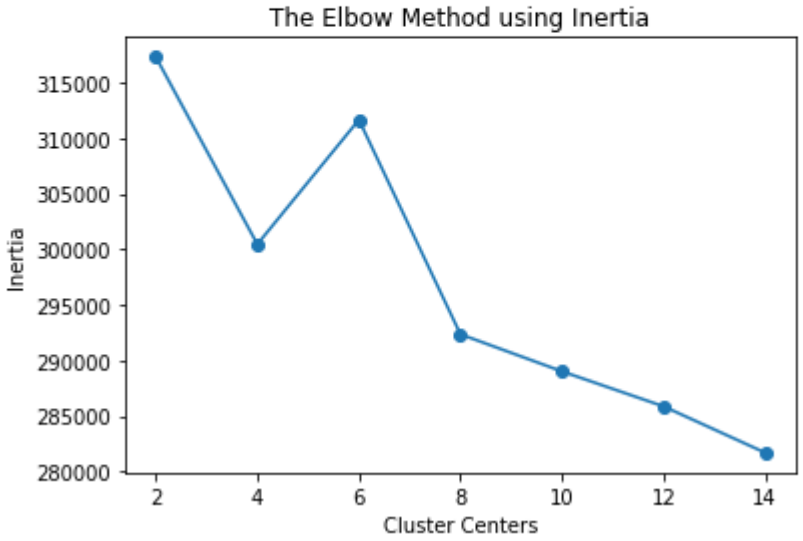
3.5. Vektorisasi TF

Vektorisasi TF

```
[ ]  # Membentuk verktorisasi IF

     tf = CountVectorizer(
          min_df = 5,
          max_df = 0.95,
          max_features = 8000,
          # stop_words = 'english'
     )
     tf.fit(data.text)
     tf_text = tf.transform(data.text)
```

```
# Mnentukan jumlah Cluster optimal menggunakan teknik Elbow pada vektorisasi TF

def find_optimal_clusters(data, max_k):
    K = range(2, max_k+1, 2)

    inertias = []

    mapping1 = {}

    for k in K:
        #tf_kMeans =
        inertias.append(MiniBatchKMeans(n_clusters=k, init_size=1024, batch_size=2048, random_state=20).fit(data).inertia_)
        print('Fit {} clusters'.format(k))

    f, ax = plt.subplots(1, 1)
    ax.plot(K, inertias, marker='o')
    ax.set_xlabel('Cluster Centers')
    ax.set_xticks(K)
    ax.set_xticklabels(K)
    ax.set_ylabel('Inertia')
    ax.set_title('The Elbow Method using Inertia')

find_optimal_clusters(tf_text, 14)
```

```
Fit 2 clusters
Fit 4 clusters
Fit 6 clusters
Fit 8 clusters
Fit 10 clusters
Fit 12 clusters
Fit 14 clusters
```



DISINI ADA GRAFIK ELBOW DENGAN VEKTORISASI TF UNTUK MENGETAHUI JUMLAH CLUSTER OPTIMAL YANG DAPAT DIGUNAKAN PADA DATA HATE SPEECH

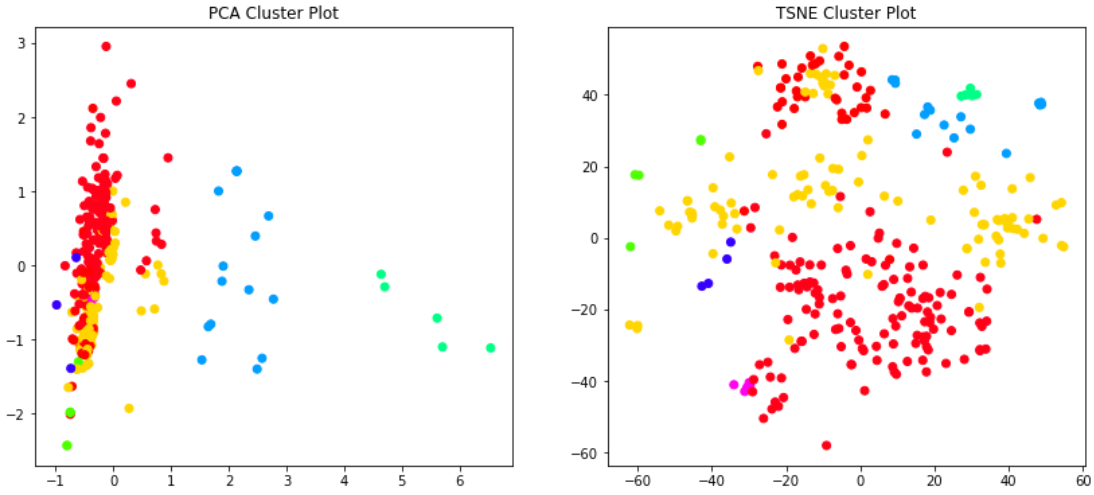
Untuk menentukan jumlah cluster yang optimal, kita pilih nilai k pada grafik Elbow yaitu titik setelah inersia mulai menurun secara linier. Jadi untuk data ini dengan vektorisasi TF, kami menggunakan jumlah cluster yang optimal adalah 8.

NEXT

```
# Membentuk model Clustering Mini Batch K-Means vektorisasi TF
# Jumlah cluster = 8
# Ukuran batch = 2048

tf_clusters = MiniBatchKMeans(n_clusters=8, init_size=1024, batch_size=2048, random_state=20).fit_predict(tf_text)
tf_clusters
```

```
array([7, 1, 1, ..., 0, 0, 1], dtype=int32)
```

```python
# Visualisasi cluster

def plot_tsne_pca(data, labels):
    max_label = max(labels)
    max_items = np.random.choice(range(data.shape[0]), size=3000, replace=False)

    pca = PCA(n_components=2).fit_transform(data[max_items,:].todense())
    tsne = TSNE().fit_transform(PCA(n_components=50).fit_transform(data[max_items,:].todense()))


    idx = np.random.choice(range(pca.shape[0]), size=300, replace=False)
    label_subset = labels[max_items]
    label_subset = [cm.hsv(i/max_label) for i in label_subset[idx]]

    f, ax = plt.subplots(1, 2, figsize=(14, 6))

    ax[0].scatter(pca[idx, 0], pca[idx, 1], c=label_subset)
    ax[0].set_title('PCA Cluster Plot')

    ax[1].scatter(tsne[idx, 0], tsne[idx, 1], c=label_subset)
    ax[1].set_title('TSNE Cluster Plot')

plot_tsne_pca(tf_text, tf_clusters)
```

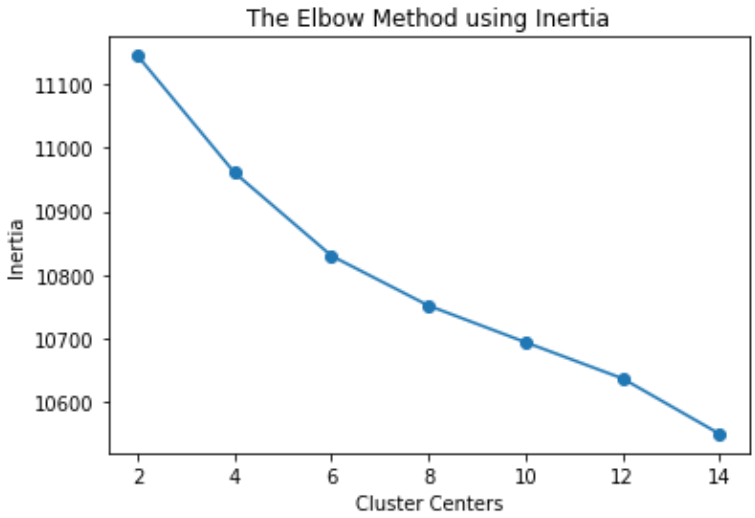


3.6. Vektorisasi TF-IDF

VEKTORISASI TF-IDF

```
[ ]  # Membentuk verktorisasi TF-IDF

     tfidf = TfidfVectorizer(
         min_df = 5,
         max_df = 0.95,
         max_features = 8000,
         # stop_words = 'english'
     )
     tfidf.fit(data.text)
     tfidf_text = tfidf.transform(data.text)
```

```
# Mnentukan jumlah Cluster optimal menggunakan teknik Elbow pada vektorisasi TF-IDF

def find_optimal_clusters(data, max_k):
    K = range(2, max_k+1, 2)

    inertias = []


    for k in K:
        #tf_kMeans =
        inertias.append(MiniBatchKMeans(n_clusters=k, init_size=1024, batch_size=2048, random_state=20).fit(data).inertia_)
        print('Fit {} clusters'.format(k))

    f, ax = plt.subplots(1, 1)
    ax.plot(K, inertias, marker='o')
    ax.set_xlabel('Cluster Centers')
    ax.set_xticks(K)
    ax.set_xticklabels(K)
    ax.set_ylabel('Inertia')
    ax.set_title('The Elbow Method using Inertia')

find_optimal_clusters(tfidf_text, 14)
```



DISINI ADA GRAFIK ELBOW DENGAN VEKTORISASI TF-IDF UNTUK MENGETAHUI JUMLAH CLUSTER OPTIMAL YANG DAPAT DIGUNAKAN

Untuk menentukan jumlah cluster yang optimal, kita pilih nilai k pada grafik Elbow yaitu titik setelah inersia mulai menurun secara linier. Jadi untuk data ini dengan vektorisassi TF-IDF, kami menggunakan jumlah cluster yang optimal adalah 6.

NEXT

```
# Membentuk model Clustering Mini Batch K-Means vektorisasi TF-IDF
# Jumlah cluster = 6
# Ukuran batch = 2048

tfidf_clusters = MiniBatchKMeans(n_clusters=6, init_size=1024, batch_size=2048, random_state=20).fit_predict(tfidf_text)
tfidf_clusters
```

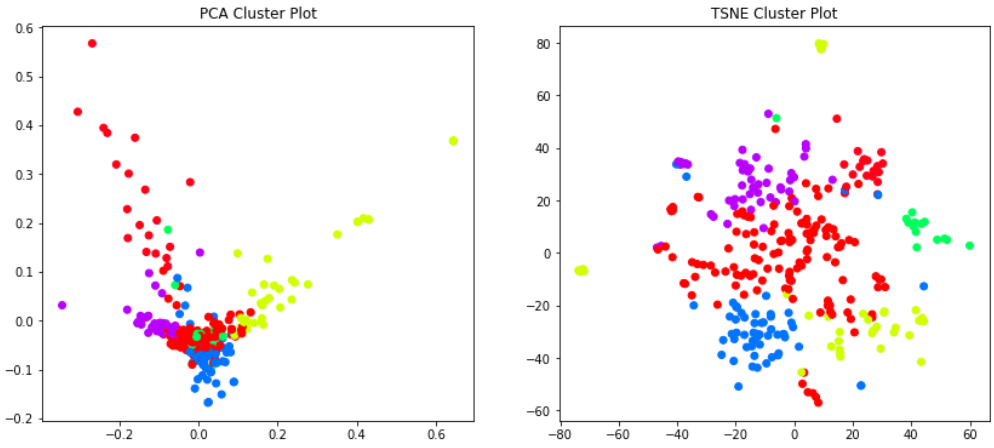
```
array([0, 4, 0, ..., 3, 3, 3], dtype=int32)
```

```
# Visualisasi cluster

def plot_tsne_pca(data, labels):
    max_label = max(labels)
    max_items = np.random.choice(range(data.shape[0]), size=3000, replace=False)

    pca = PCA(n_components=2).fit_transform(data[max_items,:].todense())
    tsne = TSNE().fit_transform(PCA(n_components=50).fit_transform(data[max_items,:].todense()))


    idx = np.random.choice(range(pca.shape[0]), size=300, replace=False)
    label_subset = labels[max_items]
    label_subset = [cm.hsv(i/max_label) for i in label_subset[idx]]

    f, ax = plt.subplots(1, 2, figsize=(14, 6))

    ax[0].scatter(pca[idx, 0], pca[idx, 1], c=label_subset)
    ax[0].set_title('PCA Cluster Plot')

    ax[1].scatter(tsne[idx, 0], tsne[idx, 1], c=label_subset)
    ax[1].set_title('TSNE Cluster Plot')

plot_tsne_pca(tfidf_text, tfidf_clusters)
```

3.7. Cluster Optimal

Berdasarkan clustering Mini Batch k-Means dengan membandingkan vektorisasi TF dan TF-IDF menggunakan teknik Elbow, diperoleh jumlah cluster sebanyak 8 cluster untuk vektorisasi TF, dan 6 cluster untuk vektorisasi TF-IDF. Oleh karena itu, kami memilih model dengan jumlah cluster yang lebih kecil yaitu 6 cluster dengan vektorisasi TF-IDF. Seperti visualisasi yang terlihat di gambar berikut.

UNTUK SELANJUTNYA AKAN DIJELASKAN OLEH REKAN SAYA



4. Klasifikasi

4.1. Grid Search CV

Dengan menggunakan grid search cv ini di peroleh parameter k yang akan digunakan pada klasifikasi ini sebanyak k =13. Dengan k=13 ini akan mengelompokkan data hate speech dan non hate menjadi 13 kelompok. Untuk evaluasi terhadap model ini akan dijelaskan pada soal berikutnya.

```
knn = KNeighborsClassifier()
from sklearn.model_selection import GridSearchCV
k_range = list(range(1, 31))
param_grid = dict(n_neighbors=k_range)

# defining parameter range
grid = GridSearchCV(knn, param_grid, cv=10, scoring='accuracy', return_train_score=False,verbose=1)

# fitting the model for grid search
grid_search=grid.fit(X_train_cv,y_train)
```

```
Fitting 10 folds for each of 30 candidates, totalling 300 fits
```

```
print(grid_search.best_params_)
```

```
{'n_neighbors': 13}
```

eroleh parameter k yang baik adalah 13

```
from sklearn.neighbors import KNeighborsClassifier
klasifikasi = KNeighborsClassifier(n_neighbors=13)
klasifikasi.fit(X_train_cv,y_train)
```

```
KNeighborsClassifier(n_neighbors=13)
```

```
y_pred = klasifikasi.predict(X_test_cv)
```

4.4 Metric evaluasi

Metric evaluasi yang digunakan pada klasifikasi k-means ini adalah confusion metric.

```
[[1239  318]
 [ 431 1462]]
              precision    recall  f1-score   support

           H       0.74      0.80      0.77      1557
           N       0.82      0.77      0.80      1893

    accuracy                           0.78      3450
   macro avg       0.78      0.78      0.78      3450
weighted avg       0.79      0.78      0.78      3450
```

hasil akurasi lebih baik

Hasil akurasi dari model k-means dengan k=13 diperoleh sebesar 78% yang artinya model sudah baik dalam memprediksi ujaran kebencian , sebelumnya kami gunakan k=3 namun akurasi yang dihasilkan tidak lebih besar dari