TUGAS AKHIR – KS184822

PERBANDINGAN METODE K-*MEANS* DAN FUZZY *C-MEANS* UNTUK PENGELOMPOKAN *TWEET* YANG DITUJUKAN KEPADA PT. KERETA API INDONESIA (@KAI121) DENGAN MENGGUNAKAN *N-GRAM*

FARIZAH RIZKA RAHMANIAR
NRP 062115 4000 0111

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

TUGAS AKHIR – KS184822

PERBANDINGAN METODE K-*MEANS* DAN FUZZY *C-MEANS* UNTUK PENGELOMPOKAN *TWEET* YANG DITUJUKAN KEPADA PT. KERETA API INDONESIA (@KAI121) DENGAN MENGGUNAKAN *N-GRAM*

FARIZAH RIZKA RAHMANIAR
NRP 062115 4000 0111

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si.

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

FINAL PROJECT – KS184822

COMPARISON OF K-MEANS AND FUZZY C-MEANS METHOD FOR GROUPING TWEET MENTIONED TO PT. KERETA API INDONESIA (@KAI121) USING N-GRAM

FARIZAH RIZKA RAHMANIAR
NRP 062115 4000 0111

Supervisor
Dr. Dra. Kartika Fithriasari, M.Si.

UNDERGRADUATE PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF MATHEMATICS, COMPUTING, AND DATA SCIENCE
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

LEMBAR PENGESAHAN

PERBANDINGAN METODE *K-MEANS* DAN *FUZZY C-MEANS* UNTUK PENGELOMPOKAN *TWEET* YANG DITUJUKAN KEPADA PT. KERETA API INDONESIA (@KAI121) DENGAN MENGGUNAKAN *N-GRAM*

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember

Oleh :
Farizah Rizka Rahmaniar
NRP. 062115 4000 0111

Disetujui oleh Pembimbing:
Dr. Dra. Kartika Fithriasari, M.Si. ( )
NIP. 19691212 199303 2 002

Mengetahui,
Kepala Departemen Statistika

Dr. Suhartono
NIP. 19710929 199512 1 001

SURABAYA, JULI 2019

PERBANDINGAN METODE *K-MEANS* DAN *FUZZY C-MEANS* UNTUK PENGELOMPOKAN *TWEET* YANG DITUJUKAN KEPADA PT. KERETA API INDONESIA (@KAI121) DENGAN MENGGUNAKAN *N-GRAM*

Nama Mahasiswa	**:**	**Farizah Rizka Rahmaniar**
NRP	**:**	**062115 4000 0111**
Departemen	**:**	**Statistika**
Dosen Pembimbing	**:**	**Dr. Dra. Kartika Fithriasari, M.Si.**

Abstrak

Banyaknya penumpang yang menggunakan jasa transportasi kereta api membuat PT. Kereta Api Indonesia harus dapat membuat kebijakan agar dapat meningkatkan kualitas pelayanannya. Salah satu cara agar dapat membuat kebijakan yang baik adalah dengan menggunakan teknologi informasi yang cepat seperti Twitter. Untuk memudahkan perusahaan agar dapat menyimpulkan suatu informasi yang bergerak dengan cepat dapat menggunakan pengelompokan tweet. Sebelum data yang diperoleh dari Twitter API tersebut diolah, maka perlu dilakukan text preprocessing seperti cleansing, case folding, stemming, normalisasi kata, stopwords removal, dan tokenizing. Selanjutnya, data akan diboboti dengan menggunakan TF-IDF. Pada penelitian ini dilakukan perbandingan metode clustering K-means dan Fuzzy C-means dengan menggunakan N-gram atau tanpa menggunakan N-gram. Clustering menggunanakan K-means dengan trigram merupakan metode terbaik untuk mengelompokkan tweet yang ditujukan kepada PT. Kereta Api Indonesia (@KAI121) karena menghasilkan nilai Calinski-Harabasz Index (CHI) yang tinggi dibandingkan metode lainnya. Hasil clustering memperoleh tiga kategori tweet yaitu tiket go-show, tiket lokal, dan tiket prameks.

Kata kunci: CHI, Clustering, Fuzzy C-means, K-means, N-gram, PT. Kereta Api Indonesia, Twitter

(Halaman ini sengaja dikosongkan)

COMPARISON OF K-MEANS AND FUZZY C-MEANS METHOD FOR GROUPING TWEET MENTIONED TO PT. KERETA API INDONESIA (@KAI121) USING N-GRAM

Name	**:**	**Farizah Rizka Rahmaniar**
Student Number	**:**	**062115 4000 0111**
Department	**:**	**Statistics**
Supervisor	**:**	**Dr. Dra. Kartika Fithriasari, M.Si.**

Abstract

The large number of passengers used train public transportation services makes PT. Kereta Api Indonesia must be able to create policy for improve the services quality. Now there are social media like Twitter that present various information as the basic information to create policy. In order to conclude information that moves quickly, crawling and grouping tweets will make it easier for companies. Before processed the data that obtained from Twitter API, it is necessary to do text preprocessing such as cleansing, case folding, stemming, word normalization, stopwords removal, and tokenizing. Furthermore, the data will be weighted using TF-IDF. In this reasearch clustering method using K-means and Fuzzy C-means was compared using N-gram or without using N-gram. Clustering using K-means method with trigram is the best method for grouping tweets mentioned to PT. Kereta Api Indonesia (@KAI121) because has a high value of the Calinski-Harabasz Index (CHI) compared to other methods. The clustering results obtained three categories of tweets. First category is go-show tickets, then local tickets, and prameks tickets.

Keywords: CHI, Clustering, Fuzzy C-means, K-means, N-gram, PT. Kereta Api Indonesia, Twitter

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan atas rahmat dan hidayah yang diberikan Allah SWT sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "Perbandingan Metode *K-means* dan *Fuzzy C-means* untuk pengelompokan *Tweet* yang Ditujukan Kepada PT. Kereta Api Indonesia (@KAI121) dengan menggunakan *N-gram*" dengan lancar.

Penulis menyadari bahwa Tugas Akhir ini dapat terselesaikan tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, atas segala do'a, nasehat, kasih sayang, dan dukungan yang diberikan kepada penulis demi kesuksesan dan kebahagiaan penulis.
2. Dr. Suhartono selaku Ketua Departemen Statistika dan Dr. Santi Wulan Purnami, S.Si., M.Si. selaku Ketua Program Studi Sarjana yang telah memberikan fasilitas, sarana, dan prasarana.
3. Dr. Drs. I Nyoman Latra, MS., Dr. Suhartono, dan Dr. Agens Tuti Rumiati, M.Sc. selaku dosen-dosen yang menjadi dosen wali selama masa studi yang telah banyak memberikan saran dan arahan dalam proses belajar di Departemen Statistika.
4. Dr. Dra. Kartika Fithriasari, M.Si. selaku dosen pembimbing yang telah meluangkan waktu dan dengan sangat sabar memberikan bimbingan, saran, dukungan serta motivasi selama penyusunan Tugas Akhir.
5. Dra. Wiwiek Setya Winahju, M.S. dan Pratnya Paramitha Oktaviana, S.Si., M.Si. selaku dosen penguji yang selalu sabar dalam mengomentari serta memberikan masukan dan saran dalam penyelesaian Tugas Akhir.
6. Seluruh dosen Statistika ITS yang telah memberikan ilmu dan pengetahuan yang tak ternilai harganya, serta segenap karyawaan Departemen Statistika ITS.
7. Teman-teman Statistika ITS ∑26 angkatan 2015, yang selalu memberikan dukungan kepada penulis selama ini.

8. Semua teman, relasi dan berbagai pihak yang tidak bisa penulis sebutkan namanya satu persatu yang telah membantu dalam penulisan laporan ini.

Besar harapan penulis untuk mendapatkan kritik dan saran yang membangun sehingga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak yang terkait.

Surabaya, Mei 2019

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Transportasi sangat berpengaruh terhadap segala aspek kehidupan manusia, seperti berperan penting dalam roda perekonomian maupun pembangunan nasional di Indonesia. Semakin meningkatnya pertumbuhan penduduk di Indonesia maka semakin meningkat pula kebutuhan-kebutuhan yang harus dipenuhi oleh masyarakat di Indonesia. Hal ini juga berdampak kepada mobilitas penduduk di Indonesia seperti pengangkutan barang maupun manusia. Transportasi memiliki pengaruh yang sangat besar dalam aspek ekonomi. Perekonomian yang semakin berkembang kearah globalisasi membutuhkan mobilitas yang tinggi, sehingga masyarakat akan semakin teliti dalam memilih sarana transportasi yang akan digunakan (Isnan, 2013). Salah satu mobilitas masyarakat yang sering digunakan adalah dengan menggunakan transportasi darat. Transportasi darat memiliki 2 jenis angkutan yaitu angkutan jalan raya seperti mobil, motor, bus, dan lainnya serta angkutan jalan rel/ kereta api.

PT. Kereta Api Indonesia (Persero) yang selanjutnya disingkat sebagai PT. KAI adalah Badan Usaha Milik Negara (BUMN) yang menyediakan, mengatur, dan mengurus jasa angkutan kereta api di Indonesia. Bisnis Perusahaan atau Pelayanan yang disediakan oleh PT. Kereta Api Indonesia diantaranya adalah angkutan penumpang, angkutan barang, dan pengusahaan aset (PT. Kereta Api Indonesia, 2016). Menurut Badan Pusat Statistik (BPS) Indonesia jumlah penumpang PT. Kereta Api Indonesia pada bulan Juli 2018 mencapai 36,8 juta penumpang (Badan Pusat Statistik, 2018). Selain itu, beberapa tahun belakangan ini PT. Kereta Api Indonesia merupakan salah satu perusahaan yang sukses mengimplementasikan *digital transformation* dimana pelayanan dan fasilitas jauh lebih berkembang dan bagus pada saat ini. Berdasarkan hal tersebut, PT. Kereta Api Indonesia harus terus dapat meningkatkan kualitas pelayanan agar penumpang dapat merasa puas terhadap pelayanan

dan dapat menjadi pelanggan yang loyal. Maka dari itu, PT. Kereta Api Indonesia harus dapat membuat kebijakan yang dapat meningkatkan kepuasan pelanggan terhadap pelayanan yang disediakan. Salah satu alat yang dapat digunakan untuk menentukan kebijakan adalah melalui penarikan suatu informasi.

Saat ini pemanfaatan teknologi informasi berkembang begitu pesatnya. Maka dari itu pemanfaatan teknologi informasi yang tepat dapat dijadikan sebagai salah satu senjata *strategic* dalam memberikan nilai tambah dalam persaingan bisnis, tak terkecuali pada PT. Kereta Api Indonesia (Wijaya & Sensuse, 2011). Teknologi informasi yang dapat digunakan untuk penarikan informasi adalah melalui sosial media dimana perusahaan dapat berinteraksi dengan pelanggan. Salah satu media sosial yang saat ini diminati oleh masyarakat Indonesia adalah *twitter* dengan menduduki peringkat ketujuh dari dua belas media sosial yang disediakan yaitu sebesar 27% (Databooks, 2018). Pengajuan keluhan, kritik, saran, maupun pertanyaan dapat disampaikan melalui *twitter*. PT. Kereta Api Indonesia memiliki akun resmi *twitter* dengan *username* @KAI121 dengan jumlah pengikut yang tergolong cukup besar. Dalam satu hari, *tweet* yang ditujukan kepada PT. Kereta Api Indonesia mencapai 500 hingga 800 *tweet* dan cukup aktif berinteraksi dengan pelanggannya. Cepatnya informasi yang masuk kepada akun *twitter* PT. Kereta Api Indonesia mengakibatkan sulitnya perusahaan untuk menarik kesimpulan dari seluruh *tweet* yang ditujukan kepada perusahaan tersebut. *Tweet* yang ditujukan kepada PT. Kereta Api Indonesia memiliki aliran informasi yang sangat cepat. Salah satu cara untuk menarik kesimpulan pada cepatnya informasi adalah dengan menggunakan *text clustering*. *Text clusteing* merupakan pengelompokan suatu data *tweet* ke dalam beberapa kelompok. Salah satu metode *text clustering* yang cukup terkenal adalah dengan menggunakan K-*means clustering* dimana metode ini mempunyai kelebihan yakni mudah untuk digunakan dalam berbagai tipe data. Namun, metode ini juga mempunyai kelemahan yaitu secara tegas mengalokasikan data ke dalam *cluster* tertentu sehingga perpindahan data dari *cluster* satu ke *cluster* lainnya tidak

dianggap. Untuk mengatasi hal tersebut terdapat metode dimana data tidak secara tegas dikelompokkan pada suatu kelompok tertentu dengan menggunakan derajat keanggotaan dan cukup efektif untuk meningkatkan homogenitas setiap *cluster* yang dihasilkan. Metode tersebut adalah Fuzzy C-*means*. Sebelum data dapat diolah, data *tweet* yang diperoleh harus diproses terlebih dahulu menggunakan *text preprocessing*. *Text preprocessing* meliputi *cleansing*, *case folding*, *stopwords removal*, *tokenizing*, dan *stemming*. Salah satu metode *tokenizing* yang dapat digunakan adalah dengan menggunakan *N-gram* dimana metode ini dapat memecah kata menjadi satu, dua, ataupun tiga kata sehingga dapat lebih merepresentasikan makna dari kalimat dalam sebuah *tweet*.

Penelitian-penelitian sebelumnya sudah pernah dilakukan oleh Deepa B. Patil dan Yashwant V. Dongre tentang kategorisasi dokumen menggunakan pendekatan fuzzy. Berdasarkan hasil penelitian tersebut pendekatan fuzzy menghasilkan *clustering* yang lebih baik dibanding dengan *hard cluster* (Patil & Dongre, 2015). Penelitian lain juga pernah dilakukan oleh Syaifudin Karyadi, Hasbi Yasin, dan Moch. Abdul Mukid tentang analisis kecenderungan informasi dengan menggunakan metode *text mining*. Penelitian ini bertujuan untuk mengetahui kecenderungan topik pemberitaan dan mengetahui topik yang paling sering muncul dari akun *twitter* @detikcom dengan menggunakan metode K-*means clustering* (Karyadi, dkk., 2016). Penelitian serupa juga dilakukan oleh Sa'idah Zahrotul Jannah yang meneliti tentang *clustering* dan visualisasi aspirasi masyarakat Surabaya dengan menggunakan *text mining*, penelitian ini menggunakan metode *clustering* K-*means* (Jannah, 2018). Penelitian K-*emans clustering* dengan data statasioner dan non-stasioner *time* series berdasarkan jarak autokorelasi dengan hierarki dan algoritma K-*means* juga pernah dilakukan yang mana hasilnya didapatkan bahwa metode K-*means* memiliki nilai akurasi paling tinggi dibanding metode hierarki (Riyadi, dkk., 2017). Selain itu penelitian tentang *text analytics* dengan menggunakan *N-gram* juga pernah dilakukan oleh Shaugi Chasbullah, dimana hasil penelitiannya menunjukkan bahwa metode *N-gram* me-

rupakan fitur yang cukup baik untuk digunakan dalam *text analytics* (Chasbullah, 2018).

Sehingga pada penelitian ini, peneliti akan melakukan perbandingan metode K-*means* dan Fuzzy *C-means* dengan menggunakan *N-gram* untuk menentukan kategori *tweet* yang ditujukan kepada PT. Kereta Api Indonesia dimana penentuan jumlah *cluster* diukur menggunakan Calinski-Harabasz *Index* (CHI) karena berdasarkan penelitian yang dilakukan oleh Milligan dan Cooper metode Calinski-Harabasz *Index* (CHI) merupakan metode terbaik untuk menentukan jumlah *cluster* (Calinski & Harabasz, 1974). Dari penelitian ini diharapkan dapat menjadi bahan rekomendasi kepada PT. Kereta Api Indonesia dalam pengelompokan *tweet* dan dapat dengan mudah menarik kesimpulan dari cepatnya informasi yang masuk. Hal tersebut agar dapat meningkatkan pelayanan terhadap pelanggan.

1.2 Rumusan Masalah

Beberapa tahun belakangan ini PT. Kereta Api Indonesia adalah salah satu perusahaan Badan Usaha Milik Negara (BUMN) yang sukses mengimplementasikan *digital transformation* dengan menyediakan pelayanan dan fasilitas yang jauh lebih berkembang dibandingkan dengan pelayanan maupun fasilitas yang dulu. Oleh karena itu, PT. Kereta Api Indonesia harus dapat mempertahankan kualitas pelayanan dan fasilitasnya agar dapat menjadi lebih baik dengan membuat beberapa kebijakan. Salah satu cara untuk membuat kebijakan adalah dengan penarikan suatu informasi di sosial media yaitu *twitter*. *Tweet* yang ditujukan kepada PT. Kereta Api Indonesia memiliki aliran informasi yang sangat cepat. Salah satu cara untuk menarik kesimpulan pada cepatnya informasi adalah dengan menggunakan *text clustering*. Metode *text clustering* yang digunakan adalah dengan menggunakan K-*means clustering* dan Fuzzy C-*means*. Data yang disediakan merupakan data teks sehingga perlu dilakukan *preprocessing* terlebih dahulu. Pada tahapan *tokenizing* digunakan metode *N-gram* karena dapat lebih merepresentasikan makna dari kalimat dalam sebuah *tweet*.

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut.

1. Mendeskripsikan data *tweet* yang ditujukan kepada PT. Kereta Api Indonesia.
2. Mengelompokan *tweet* yang ditujukan kepada PT. Kereta Api Indonesia menggunakan metode K-*means* dan Fuzzy *C-means* dimana tahapan *preprocessing* menggunakan *N-gram*.
3. Memvisualisasikan *tweet* yang ditujukan kepada PT. Kereta Api Indonesia.

1.4 Manfaat Penelitian

Manfaat penelitian ini adalah dapat menjadi bahan rekomendasi bagi PT. Kereta Api Indonesia untuk mengelompokkan informasi yang bergerak dengan cepat seperti Twitter guna meningkatkan kualitas pelayanan.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah data yang digunakan hanya *tweet* yang ditujukan kepada akun PT. Kereta Api Indonesia (@KAI121). Selain itu *tokenizing* hanya menggunakan unigram, bigram, dan trigram.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bagian ini akan membahas beberapa kajian pustaka yang digunakan untuk menyelesaikan permasalahan mengenai pengelompokan *tweet* yang ditujukan kepada PT. Kereta Api Indonesia.

2.1 *Text* Mining

Text Mining merupakan suatu proses untuk mengekstrak pola yang menarik untuk mengeksplorasi pengetahuan dari sumber data tekstual (Weiguo, dkk., 2016). *Text mining* juga dapat digunakan untuk menganalisis dari suatu hal yang menarik dan dapat mendapatkan informasi relevan secara efektif dan efisien dari sejumlah besar *unstructured data* (Talib, dkk., 2016). *Text mining* meliputi kategorisasi teks, pengelompokan teks (*clustering*), ekstraksi konsep/entitas, analisis sentimen, dan sebagainya (Han, dkk., 2012).

Text mining memiliki data yang berupa teks dimana tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisis. Tahapan *text mining* yang dilakukan pada penelitian ini adalah *text preprocessing*, *text clustering*, evaluasi *clustering* menggunakan Calinski-Harabasz *Index* (CHI), dan visualisasi.

2.2 *Text Preprocessing*

Text Preprocessing merupakan tahap awal yang dilakukan dalam *text mining* dimana data teks dipersiapkan agar dapat diolah lebih lanjut (Feldman & Sanger, 2007). *Preprocessing* merupakan sebuah proses pengurangan kata-kata yang tidak penting atau tidak mempunyai arti dari *database* teks, sehingga data dapat lebih terstruktur dan siap untuk diolah (Harjanta, 2015). Ada beberapa tahapan dalam *text preprocessing* yaitu sebagai berikut.

1. *Cleansing* dimana proses ini merupakan proses memperkecil *noise* yang dapat dilakukan dengan menghilangkan data yang salah, memperbaiki kekacauan data, dan memeriksa data yang tidak konsisten

2. *Case Folding* merupakan proses untuk merubah semua huruf menjadi huruf kecil. Huruf yang dapat dihapus hanya huruf "a" sampai dengan "z". Karakter selain huruf tersebut akan dihilangkan dan dianggap sebagai delimiter (Feldman & Sanger, 2007).
3. *Stopwords removal* merupakan proses untuk mengambil kata-kata yang penting. Kata-kata yang dianggap tidak penting dapat dibuang dengan menggunakan *stopwords* (Feldman & Sanger, 2007).

2.2.1 *Tokenizing*

Tokenizing merupakan tahap pemotongan kata pada satu kalimat. *Tokenizing* yang dilakukan dapat menggunakan dengan metode *N-gram*. *Tokenizing N-gram* merupakan pemotongan kata dengan melihat urutan karakter atau kata yang diekstrak dari sebuah teks. *N-gram* dapat dibagi menjadi dua kategori yaitu berdasarkan karakter dan berdasarkan kata. Karakter *N-gram* adalah sekumpulan *n* karakter berurutan yang diekstrak dari sebuah kata. Motivasi utama dibalik pendekatan ini adalah bahwa kata-kata yang sama akan memiliki proporsi *N-gram* yang tinggi (Majumder., dkk., 2002). Pada penelitian ini *N-gram* yang digunakan adalah berdasarkan pasangan kata yang dipotong dari awal sampai akhir dokumen. Berikut merupakan contoh rangkaian *N-gram* dari kalimat "saya naik kereta api".

Tabel 2.1 Contoh Rangkaian *N-gram*

	N	Kata *N-gram*
Unigram	1	Saya
	1	Naik
	1	Kereta
	1	Api
Bigram	2	Saya Naik
	2	Naik Kereta
	2	Kereta Api
Trigram	3	Saya Naik Kereta
	3	Naik Kereta Api

2.2.2 *Stemming*

Stemming merupakan proses pemotongan kata-kata menjadi kata dasarnya. Pada penelitian ini menggunakan proses *stemming* menggunkaan salah satu *library* dari Python yang dinamakan Sastrawi. Sastrawi dapat digunakan untuk proses *stemming* untuk dokumen berbahasa Indonesia. Sastrawi merupakan salah satu *stemmer* yang dibuat agar mudah untuk digunakan. *Stemmer* ini merupakan pengembangan dari algoritma dari Nazief dan Adriani atau dapat disebut dengan algorima *confix-stripping stemmer*. Berdasarkan penggunaan afiks, berikut merupakan model kata dalam bahasa Indonesia (Adriani, dkk., 2007).

[[[DP+]DP+]DP+] kata dasar [[DS+][+PP][+P]]

dimana kurung siku menandakan bahwa afiks opsional, DP (*Derivational Prefixes*) adalah awalan, DS (*Derivational Suffixes*) adalah akhiran, PP (*Possesive Pronouns*) adalah kata ganti kepunyaan, dan P (*particle*) adalah partikel. Tahapan dalam *confix-stripping stemmer* dapat dilakukan sebagai berikut (Adriani, dkk., 2007).

1. Setiap langkah awal dalam pemrosesan adalah melakukan pemeriksaan pada kamus kata dasar. Jika kata dari proses *stemming* ditemukan dalam kamus, maka kata dianggap sebagai kata dasar dan proses *stemming* dapat dihentikan.
2. Menghilangkan *inflectional suffixes* yang dapat dimulai dari menghapus *particle* (P) seperti {-kah, -lah, -tah, atau, -pun}, lalu menghapus *possessive pronoun* (PP) {-ku, -mu, -nya}. Misal seperti "bukumu" akan dipotong menjadi "bukumu" dan kemudian "buku" lalu *stemming* akan berhenti karena kata tersebut merupakan kata dasar.
3. Menghilangkan *derivational suffixes* seperti {-i, -kan, -an}. Contohnya seperti kata "membelikan" maka kata akan dipotong menjadi "membeli", tetapi kata "membeli" bukanlah kata dasar, maka dilanjutkan ke langkah selanjutnya.
4. Menghilangkan *derivational prefixes* {be-, di-, ke-, se-, me-, te-, pe-}.

a. Proses berhenti jika:

- Awalan yang diidentifikasi membentuk pasangan afiks dengan

akhiran yang terlarang yang telah dihilangkan pada langkah 3. Pasangan awalan akhiran yang dilarang seperti pada tabel berikut.

Tabel 2.2 Pasangan Awalan dan Akhiran yang Dilarang

Awalan	Akhiran yang dilarang
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

- Awalan yang diidentifikasi identik dengan awalan yang sebelumnya dihapus.
- Tiga awalan yang telah dihapus.

b. Identifikasi jenis awalan yang terdiri dari dua jenis.
- Standar yang dapat dihilangkan langsung dari kata seperti {di-, ke-, se-}.
- Complex yang dapat diubah sesuai kata dasar yang mengikutinya (dapat mengubah bentuk asli kata dasar) seperti {be-, te-, me-, pe-}.

c. Pencarian kata yang awalannya telah dihapus dalam kamus kata dasar, jika pencarian tidak ditemukan maka ulangi tahapan ke-4. Namun, apabila kata dapat ditemukan dalam kamus kata dasar maka keseluruhan proses dapat berhenti.

5. Jika pada langkah 4 masih belum ditemukan kata dasarnya maka dilakukan *recoding* yaitu menambah atau mengganti huruf awal kata yang terpenggal pada proses *stemming*. Contohnya adalah seperti kata “menerjang”, jika awalan “men-” dihilangkan maka akan membentuk kata “erjang”. Kata tersebut bukanlah kata dasar sehingga perlu dilakukan *recoding* menjadi kata “terjang”.
6. Jika semua langkah tidak berhasil dilakukan, maka kata tersebut

dianggap sebagai kata dasar dan algoritma akan mengembalikan kata seperti semula.

2.3 *Term Weighting*

Term weighting atau pembobotan kata dilakukan dengan menghitung frekuensi kemunculan *term* dalam dokumen. Salah satu metode *term weighting* (pembobotan kata) adalah dengan menggunakan TF-IDF. *Term Frequency Inverse Document Frequency* (TF-IDF) merupakan konsep pembobotan *term* pada suatu dokumen. *Term Frequency* (TF) merepresentasikan banyaknya *term t* yang muncul pada kalimat *d* ($tf_{t,d}$) atau dapat dikatakan menunjukkan jumlah kemunculan sebuah kata pada suatu ulasan. *Inverse Document Frequency* (IDF) merepresentasikan banyaknya kalimat yang mengandung *term t* (df_t) atau menunjukkan frekuensi kemunculan kata pada seluruh kumpulan ulasan (Hayatin, dkk., 2015). TF-IDF didapatkan dengan mengalikan nilai *Term Frequency* (TF) dengan nilai *Inverse Document Frequency* (IDF). Berikut merupakan persamaan yang membentuk TF-IDF (Manning, dkk., 2009).

$$tf - idf_{t,d} = tf_{t,d} \times \log\left(\frac{N}{df_t}\right) \tag{2.1}$$

Keterangan:
N : jumlah seluruh *tweet*
$tf_{t,d}$: jumlah kemunculan kata ke-*t* pada *tweet* ke-*d*
df_t : banyaknya *tweet* yang mengandung kata ke-*t*

2.4 *Text Clustering*

Clustering adalah proses pengelompokan suatu set objek data menjadi beberapa kelompok atau *cluster*, sehingga objek yang berada di dalam *cluster* yang sama memiliki kesamaan namun sangat berbeda dengan objek yang berada di kelompok lain (Witten & Frank, 2012). Beberapa metode untuk menemukan kesamaan objek dalam sebuah *cluster* dapat menggunakan jarak diantara mereka. Jarak seperti itu dapat didefinisikan menggunakan jarak Euclidean, persamaan cosine, Jaccard *coefficient*, dan lain sebagainya (Patil & Dongre, 2015).

Text clustering adalah aplikasi analisis *cluster* untuk dokumen berbasis teks. Ia menggunakan *machine learning* dan *natural language processing* (NLP) untuk memahami dan mengelompokkan data tekstual yang tidak terstruktur. Pada penelitian ini metode yang digunakan untuk *text clustering* adalah K-*means* dan Fuzzy *C-means*.

2.4.1 *K-Means*

K-*means* merupakan salah satu algoritma dalam data *mining* yang bisa digunakan untuk melakukan pengelompokan/*clustering* suatu data. Metode K-*means* adalah metode yang termasuk dalam algoritma *clustering* berbasis jarak yang membagi data ke dalam sejumlah *cluster* (Witten & Frank, 2012). Selain itu, K-*means* juga merupakan teknik pengelompokan non-hierarki yang dapat digunakan untuk mempartisi data kedalam *cluster* yang memiliki kedekatan dengan *centroid*. Secara umum metode K-*means* bekerja dengan langkah-langkah yang dapat dijelaskan seperti berikut (Johnson & Wichern, 2007).

1. Menentukan *K* sebagai jumlah *cluster* yang ingin dibentuk.
2. Menentukan pusat *cluster* (*centroid*). Dalam menentukan *centroid* awal dilakukan secara acak dimana data yang digunakan merupakan kata dasar dari setiap *tweet* yang telah diboboti.
3. Membentuk *K cluster* dimana setiap *tweet* akan ditempatkan dengan *centroid* terdekat. Persamaan berikut merupakan perhitungan *centroid* terdekat yang dilakukan dengan menggunakan jarak *Euclidean*.

$$D_{ij} = \sqrt{\sum_{k=1}^{p} \left(x_{kj} - y_{ki} \right)^2} \tag{2.2}$$

   Keterangan:
   D_{ij} : jarak *Euclidean* dari *tweet* ke-*j* ke pusat *cluster* ke-*i*
   x_{kj} : frekuensi kemunculan kata ke-*k* pada *tweet* ke-*j*
   y_{ki} : frekuensi kemunculan kata ke-*k* pada *cluster* ke-*i*
   p : jumlah kata
4. Menghitung kembali *centroid* setiap *cluster* untuk iterasi berikutnya menggunakan persamaan sebagai berikut.

$$v_{ik} = \frac{\sum_{j=1}^{n_i} x_{kj}}{n_i} \quad (2.3)$$

Keterangan:

v_{ik} : *centroid* (rata-rata *cluster* ke-*i* untuk kata ke-*k*)

x_{kj} : frekuensi kemunculan kata ke-*k* pada *tweet* ke-*j* yang berada pada *cluster* tersebut

n_i : jumlah data pada *cluster* ke-*i*

5. Mengulangi kembali langkah 3-4 sampai tidak ada lagi perpindahan objek atau sudah konvergen.

2.4.2 *Fuzzy C-means*

Fuzzy *Clustering* adalah salah satu teknik untuk menentukan *cluster* optimal dalam suatu ruang vektor yang didasarkan pada bentuk normal *Euclidian* untuk jarak antar vektor. Fuzzy *clustering* sangat berguna bagi pemodelan fuzzy terutama dalam mengidentifikasi aturan-aturan fuzzy.

Ada beberapa algoritma *clustering* data, salah satu diantaranya adalah Fuzzy *C-means* (FCM). Fuzzy *C-means* (FCM) adalah salah satu teknik untuk membentuk *cluster* data yang mana keberadaan tiap-tiap titik data dalam suatu *cluster* ditentukan oleh derajat keanggotaan (Kusumadewi & Prunomo, 2010). Metode Fuzzy *C-means* (FCM) merupakan salah satu metode pengelompokan yang dikembangkan dari K-Means dengan menerapkan sifat fuzzy keanggotaannya. Metode FCM mengalokasikan kembali data ke dalam masing-masing kelompok memanfaatkan teori fuzzy. Dalam metode FCM digunakan variabel derajat keanggotaan (μ_{ik}) yang merujuk pada seberapa besar kemungkinan suatu data dapat menjadi anggota dalam suatu *cluster*. Berikut merupakan langkah-langkah pada metode Fuzzy *C-means* (FCM) (Kusumadewi & Prunomo, 2010).

1. *Input* data yang akan dibuat *cluster* $\boldsymbol{x}$, berupa matriks berukuran $n \times m$ (n = jumlah *tweet*, m = jumlah kata). x_{ij} = data sampel ke-i = 1,2,…,n) dengan atribut ke-j (j=1,2,…, m).

$$\mathbf{x} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} \\ x_{21} & x_{22} & \dots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} \end{bmatrix}$$

Contoh matriks untuk *input* data dapat disajikan seperti di atas.

2. Menentukan jumlah *cluster* (*c*) berdasarkan nilai CHI tertinggi, pangkat matriks (*w*), maksimum iterasi (*MaxIter*), *error* terkecil yang diharapkan (ξ), fungsi objektif awal ($P_0 = 0$), iterasi awal ($t=1$).
3. Membangkitkan bilangan *random* untuk menentukan derajat keanggotaan μ_{ik}, i=1,2,…,n; k=1,2,…,c sebagai elemen-elemen matrik partisi awal U. Batas nilai derajat keanggotaan terletak pada interval 0 sampai 1.
4. Menghitung pusat *cluster* ke-*k* menggunakan persamaan sebagai berikut (Yan, dkk., 1994).

$$V_{kj} = \frac{\sum_{i=1}^{n}\left(\left(\mu_{ik}\right)^{w} x_{ij}\right)}{\sum_{i=1}^{n}\left(\mu_{ik}\right)^{w}} \tag{2.4}$$

Ketengan:
V_{kj} : pusat *cluster* ke-*k* untuk kata ke-*j*
μ_{ik} : matriks partisi awal untuk *tweet* ke-*i* pada *cluster* ke-*k*
x_{ij} : frekuensi kemunculan kata ke-*j* pada *tweet* ke-*i*
n : jumlah data *tweet*

5. Menghitung fungsi objektif pada iterasi ke-*t* (Yan, dkk., 1994).

$$P_t = \sum_{i=1}^{n}\sum_{k=1}^{c}\left(\left[\sum_{j=1}^{m}\left(x_{ij} - V_{kj}\right)^2\right]\left(\mu_{ik}\right)^{w}\right) \tag{2.5}$$

Ketengan:
m : jumlah kata
c : banyak *cluster*

6. Menghitung kembali perubahan matriks partisi dengan rumus seperti pada persamaan 2.6 (Yan, dkk., 1994).

$$\mu_{ik} = \frac{\left[\sum_{j=1}^{m}\left(x_{ij} - V_{kj}\right)^2\right]^{\frac{-1}{w-1}}}{\sum_{k=1}^{c}\left[\sum_{j=1}^{m}\left(x_{ij} - V_{kj}\right)^2\right]^{\frac{-1}{w-1}}} \quad (2.6)$$

Keterangan:

μ_{ik} : perubahan matriks partisi untuk *tweet* ke-*i* pada *cluster* ke-*k*

7. Memeriksa kondisi berhenti dengan beberapa kondisi sebagai berikut.

a. Jika ($|P_t - P_{t-1}| < \xi$) atau ($t < MaxIter$) maka berhenti.
b. Jika tidak maka mengulangi langkah ke-4.

2.5 Calinski-Harabasz *Index*

Jumlah cluster (K) optimum dapat dihitung menggunakan Calinski-Harabsz *Index* (CHI). Calinski-Harabasz *Index* (CHI) juga dapat disebut *Variance Ratio Criterion* (VRC) yang merupakan rasio antara variansi antar *cluster* (*Between Group Sum of Squares*/BGSS) terhadap variansi dalam *cluster* (*Within Group Sum of Squares*/WGSS). Jarak antar *cluster* (BGSS) diharapkan memiliki nilai yang tinggi, sedangkan jarak dalam *cluster* (WGSS) diharapkan kecil oleh karena itu nilai Calinski-Harabasz *Index* (CHI) dipilih nilai yang tinggi dimana menandakan hubungan yang erat dalam cluster tersebut dan memisahkan antar cluster dengan baik. Berikut merupakan persamaan untuk menghitung nilai Calinski-Harabasz *Index* (CHI) (Calinski & Harabasz, 1974).

$$\mathrm{CHI} = \frac{BGSS/K-1}{WGSS/N-K} = \frac{\left(\sum_{i=1}^{K}\sum_{l=1,l\neq i}^{K}\sum_{k=1}^{p}\left(y_{ki} - y_{kl}\right)^2\right)/K-1}{\left(\sum_{i=1}^{K}\sum_{j=1}^{n_i}\sum_{k=1}^{p}\left(x_{kj} - y_{ki}\right)^2\right)/N-K} \quad (2.7)$$

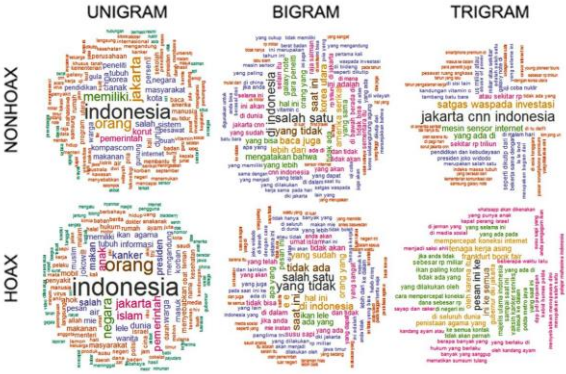
Keterangan:

K : banyak *cluster*

N : total banyak *tweet*
y_{ki} : frekuensi kemunculan kata ke-k pada *cluster* ke-i
y_{kl} : frekuensi kemunculan kata ke-k pada *cluster* ke-l
x_{kj} : frekuensi kemunculan kata ke-k pada *tweet* ke-j
n_i : banyak *tweet* pada *cluster* ke-i
p : total banyak kata

2.6 *Word cloud*

Word cloud merupakan sebuah sistem yang memunculkan visualisasi kata-kata dengan memberikan penekanan pada frekuensi kemunculan kata terkait dalam wacana tertulis. Pemakaian *word cloud* dalam analisis wacana atau *text mining* dapat memudahkan peneliti karena mampu memberikan gambaran mengenai garis besar teks dengan cepat. Aplikasi *word cloud* dilakukan untuk mendapatkan kilasan singkat tentang intisari data (McNaught & Lam, 2010). Berikut merupakan contoh *word cloud* dengan menggunakan unigram, bigram, maupun trigram.



Gambar 2.1 *Word Cloud* (Chasbullah, 2018)

2.7 *Twitter*

Twitter merupakan salah satu layanan *social networking* yang termasuk dalam *microblogging* dimana penggunanya dapat mengirim ataupun membaca pesan berbasis teks hingga 140 karakter, namun pada tanggal 07 November 2017 karakter yang dapat ditulis bertambah menjadi 280 karakter yang biasa dikenal dengan sebutan kicauan (*tweet*). Twitter dapat diakses melalui web atau

perangkat seluler. Fitur kicauan (*tweet*) yang disediakan oleh twitter dapat mengungkapkan peristiwa yang sedang terjadi di dunia dan yang dibincangkan oleh orang-orang saat ini.

Twitter juga menyediakan akses programatik ke data Twitter kepada perusahaan, pengembang, dan pengguna API (*Application Programming Interface*) guna membagikan informasi seluas mungkin. Twitter mengizinkan akses ke bagian dari layanannya melalui API agar orang-orang dapat membangun perangkat lunak yang terintegrasi dengan Twitter seperti solusi yang membantu sebuah perusahaan menjawab umpan balik pelanggan di Twitter. Salah satu *endpoint* (alamat yang terkait dengan informasi jenis tertentu yang disediakan oleh Twitter) adalah *tweet* dan balasan dimana pengembang dapat mengakses *tweets* dengan mencari kata kunci tertentu.

(Halaman ini sengaja dikosongkan)

BAB III
METODOLOGI PENELITIAN

3.1 Data

Data yang digunakan dalam penelitian ini adalah data sekunder yang didapat dengan menggunakan Twitter API (*Application Programming Interface*) selama 8 Maret 2019 hingga 18 Maret 2019. Data tersebut berupa *tweet* yang ditujukan masyarakat kepada PT. Kereta Api Indonesia (@KAI121).

Variabel penelitian yang digunakan adalah frekuensi kemunculan kata dasar setiap *tweet* yang ditujukan masyarakat kepada PT. Kereta Api Indonesia (@KAI121) yang dinotasikan dalam variabel a_{ij}. Kata dasar yang digunakan harus sudah melalui tahap *preprocessing* data.

Struktur data pada penelitian menggunakan data yang telah dilakukan *preprocessing*. Berikut merupakan contoh data sebelum dilakukan *preprocessing*.

Tabel 3.1 Contoh Data

No	***Created_at***	***Screen_name***	***Text***
1	3/8/2019	Nama 1	*Tweet* ke-1
2	3/8/2019	Nama 2	*Tweet* ke-2
⋮	⋮	⋮	⋮
4166	3/18/2019	Nama *n*	*Tweet* ke-*n*

Contoh data seperti pada Tabel 3.1 akan dilakukan *text preprocessing*, sehingga struktur data dapat dilihat pada Tabel 3.2.

Tabel 3.2 Struktur Data Setelah *Preprocessing*

***Tweet* ke-**	$a_{\cdot 1}$	$a_{\cdot 2}$	$a_{\cdot 3}$	**…**	$a_{\cdot m}$
1	a_{11}	a_{12}	a_{13}	…	a_{1m}
2	a_{21}	a_{22}	a_{23}	…	a_{2m}
⋮	⋮	⋮	⋮	⋱	⋮
4166	$a_{4166;1}$	$a_{4166;2}$	$a_{4166;3}$	…	$a_{4166;m}$

Keterangan:
n : banyak *tweet*.

m : banyak kata.
a_{ij} : kata ke-j yang sudah diboboti menggunakan TF-IDF pada *tweet* ke-i, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$.

3.2 Langkah Analisis

Langkah-langkah analisis yang dilakukan pada penelitian ini adalah sebagai berikut.

1. Mengambil data dari *twitter* menggunakan *Twitter* API dengan *keyword* @KAI121 dan disimpan dalam suatu *database*.
2. Melakukan *text preprocessing*.

a. Melakukan *cleansing* data yang meliputi menghapus *link* URL, simbol *retweet* (RT), *username* (*@username*), *emoticon*, tanda baca, simbol *hashtag* (*#hashtag*), serta simbol lain yang tidak diperlukan.
b. Melakukan *case folding*, yaitu merubah semua huruf menjadi huruf kecil.
c. Melakukan *stemming*, yaitu menghilangkan kata imbuhan untuk mendapatkan kata dasar. Proses *stemming* yang dilakukan menggunakan *package* Sastrawi yang terdapat di Python.
d. Normalisasi kata, yaitu mengubah kata-kata agar menjadi makna yang sama atau seragam.
e. Melakukan *stopwords removal*, yaitu menghapus kata-kata pada *tweet* yang terdapat pada daftar *stopwords*.
f. Melakukan *tokenizing* menggunakan *N-gram*, yaitu proses pemecahan kata menjadi unigram, bigram, dan trigram.
g. Melakukan *term weighting* atau konversi data *tweet* menjadi numerik dengan menghitung banyaknya kemunculan kata dasar serta pembobotan dengan menggunakan TF-IDF. Setelah menghitung TF dan IDF maka pembobotan suatu kata dalam suatu *tweet* dapat dihitung menggunakan persamaan (2.1).

3. Melakukan *clustering* data.

a. Melakukan *clustering* data menggunakan *K-means*.

- Menentukan jumlah *cluster* optimum dengan menghitung nilai Calinski-Harabasz *Index* (CHI) dari setiap kombinasi *K* pada data unigram, bigram, dan trigram dengan menggunakan persamaan 2.7.

- Melakukan *clustering* sesuai langkah-langkah pada subbab 2.4.1 pada jumlah *cluster* optimum dengan memilih nilai CHI tertinggi.

b. Melakukan *clustering* menggunakan Fuzzy *C-means*.

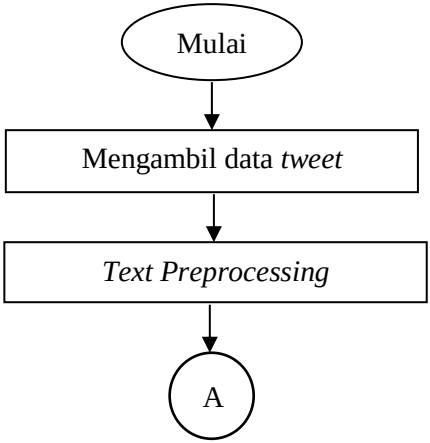
- Menentukan jumlah *cluster* optimum dengan menghitung nilai Calinski-Harabasz *Index* (CHI) dari setiap kombinasi *K* pada data unigram, bigram, dan trigram dengan menggunakan persamaan 2.7.
- Melakukan *clustering* sesuai langkah-langkah pada subbab 2.4.1 pada jumlah *cluster* optimum dengan memilih nilai CHI tertinggi.

c. Membandingkan dan memilih nilai CHI tertinggi antara metode K-*means* dan Fuzzy C-*means*.

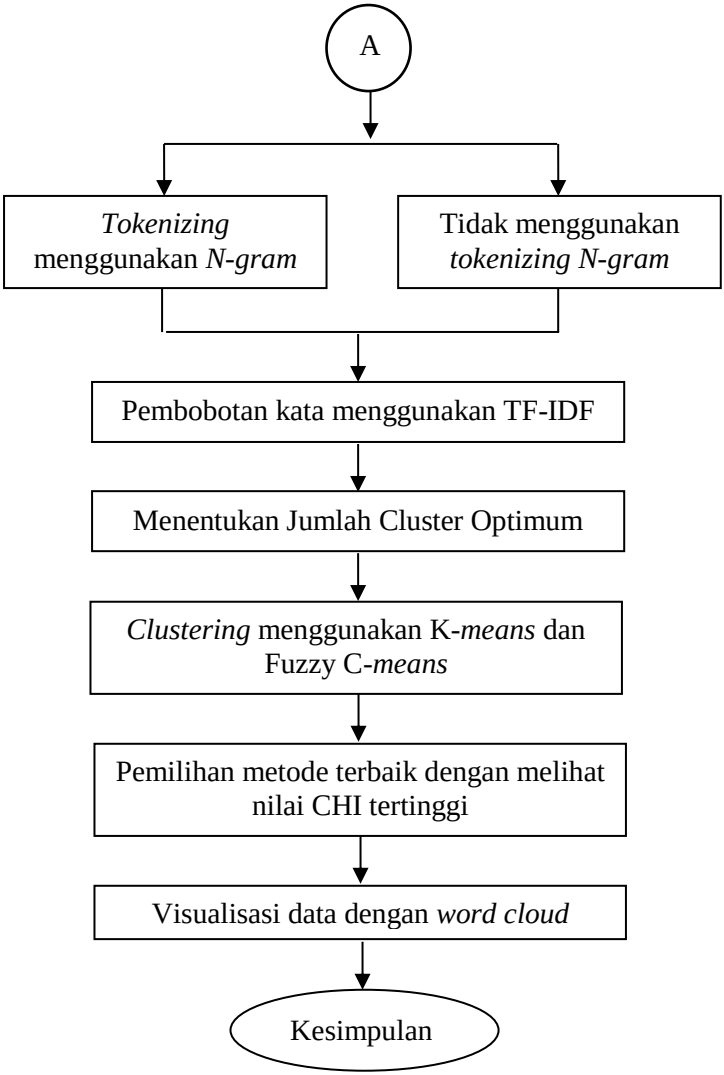
6. Membuat visualisasi data dengan *word cloud*.
7. Menarik kesimpulan dan saran.

3.3 Diagram Alir

Dalam penelitian ini langkah analisis digambarkan dalam diagram alir sebagai berikut.



Gambar 3.1 Diagram Alir Penelitian



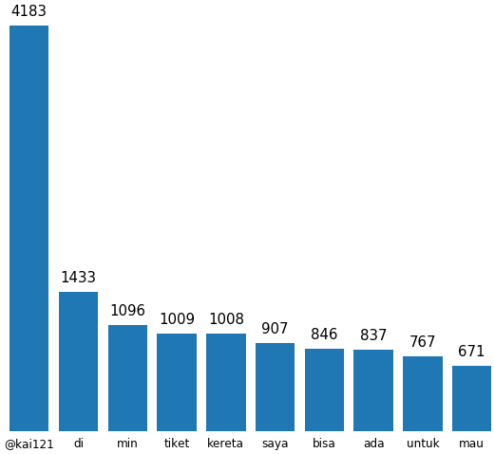
Gambar 3.2 Diagran Alir Penelitian (Lanjutan)

BAB IV
ANALISIS DAN PEMBAHASAN

Pada analisis dan pembahasan ini akan dibahas mengenai karakteristik data *tweet* yang ditujukan kepada PT. Kereta Api Indonesia (@KAI121) sebelum dan setelah tahapan *preprocessing*. Setelah itu dilakukan pengelompokan data *tweet* menggunakan metode *K-means* dan *Fuzzy C-means*. Pengelompokan data *tweet* juga dilakukan dengan menggunakan *N-gram*.

4.1 Karakteristik Data Tweet dan *Text Preprocessing*

Hasil *crawling tweet* dengan Twitter API periode 8 Maret 2019 hingga 18 Maret 2019 mendapatkan 4166 *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia. Karakteristik data *tweet* yang diperoleh dapat digambarkan dengan menggunakan diagram batang frekuensi kemunculan kata tertinggi.



Gambar 4.1 Diagram Batang Sepuluh Kata dengan Frekuensi Kemunculan Tertinggi Sebelum Preprocessing

Dari Gambar 4.1 dapat diketahui bahwa sepuluh kata yang memiliki frekuensi kemunculan tertinggi adalah “@kai121”, “di”, “min”, “tiket”, “kereta, ”“saya”, “bisa”, “ada”, “untuk”, dan “mau”. Beberapa dari sepuluh kata tersebut tidak memiliki makna

atau bukan kata baku. Sehingga, perlu untuk dilakukan *text preprocessing* yang bertujuan untuk mendapatkan data yang siap untuk di analisis menggunakan metode statistika. Tahapan *text preprocessing* yang dapat dilakukan adalah dengan *cleansing* data dimana data *tweet* akan dibersihkan dengan cara menghapus *link* URL, simbol retweet (RT), *username* (@username), *emoticon*, tanda baca, simbol hashtag (#hashtag), ataupun simbol lain yang tidak diperlukan. Setelah itu, data *tweet* diubah menjadi huruf kecil atau yang biasa disebut dengan *case folding*. Setelah semua huruf menjadi huruf kecil dilakukan proses *stemming* dimana data *tweet* akan diolah dengan mengubah kata-katanya menjadi kata dasar atau dengan kata lain menghilangkan imbuhan-imbuhan yang terdapat pada kata tersebut. Proses selanjutnya yaitu normalisasi kata atau menyeragamkan kata-kata yang memiliki makna yang sama. Lalu kata-kata yang tidak memiliki makna akan dihilangkan, proses ini disebut *stopwords removal*. Setelah semua kata-kata telah dibersihkan maka dilakukan *tokenizing* atau proses pemecahan kata. Pada penelitian ini pemecahan kata dilakukan dengan menggunakan *N-gram* dimana kata-kata tersebut akan dipecah menjadi unigram, bigram, dan trigram. Ilustrasi untuk tahapan *cleansing* sampai *tokenizing* akan ditampilkan pada Tabel 4.1 menggunakan contoh kalimat *tweet* "@KAI121 kmrn saya beli 2 tiket prameks statusnya masih "proses", hari ini saya lihat tiket tersebut hilang dan saldo Link Aja sudah terpotong, bgmn solusinya?" ditunjukkan seperti pada Tabel 4.1.

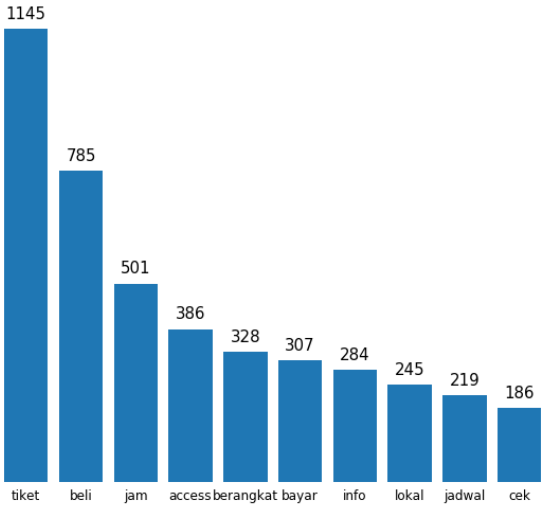
Tabel 4.1 Ilustrasi *Text Preprocessing*

Tahapan	Output	Keterangan
Sebelum *Text Preprocessing*	@KAI121 kmrn saya beli 2 tiket prameks statusnya masih "proses", hari ini saya lihat tiket tersebut hilang dan saldo Link Aja sudah terpotong, bgmn solusinya?	*Tweet* sebelum dilakukan *text preprocessing*
Cleansing	kmrn saya beli tiket prameks statusnya masih proses hari ini saya lihat tiket tersebut hilang dan saldo Link Aja sudah terpotong bgmn solusinya	Menghapus *username*, angka, dan tanda baca

Tabel 4.2 Ilustrasi *Text Preprocessing* (Lanjutan I)

Tahapan	Output	Keterangan
Case Folding	kmrn saya beli tiket prameks status-nya masih proses hari ini saya lihat ti-ket tersebut hilang dan saldo link aja sudah terpotong bgmn solusinya	"Link Aja" diubah huruf kecil semua menjadi "link aja"
Stemming	kmrn saya beli tiket prameks status masih proses hari ini saya lihat tiket sebut hilang dan saldo link aja sudah potong bgmn solusi	"terpotong" menjadi "potong" imbuhan "ter-" dihilangkan
Normalisasi Kata	kemarin saya beli tiket prameks status masih proses hari ini saya lihat tiket sebut hilang dan saldo linkaja sudah potong bgmn solusi	Singkatan "kmrn" diseragamkan menjadi "kemarin"
Stopword Removal	kemarin beli tiket prameks status pro-ses tiket hilang saldo linkaja potong solusi	Kata-kata seperti "saya", "dan", dll dihilangkan
Tokenizing Unigram	"kemarin", "beli", "tiket", "prameks", "status", "proses", "hilang", "saldo", "linkaja", "potong", "solusi"	Memecah kalimat menjadi satu kata
Tokenizing Bigram	"kemarin beli", "beli tiket", "tiket prameks", "prameks status", "status proses", "proses tiket", "tiket hilang", "hilang saldo", "saldo linkaja", "linkaja potong", "potong solusi"	Memecah kalimat menjadi dua kata
Tokenizing Trigram	"kemarin beli tiket", "beli tiket prameks", "tiket prameks status", "prameks status proses", "status proses tiket", "proses tiket hilang", "tiket hilang saldo", "hilang saldo linkaja", "saldo linkaja potong", "linkaja potong solusi"	Memecah kalimat menjadi tiga kata

Setelah dilakukan *text preprocessing* jumlah kata berkurang menjadi 2.711 kata, hal ini dikarenakan beberapa kata tidak memiliki makna yang signifikan. Karakteristik data dengan menghitung frekuensi kemunculan kata terbanyak setelah dilakukan *text preprocessing* ditampilkan pada Gambar 4.2. Pada Gambar 4.2 menunjukkan adanya perubahan frekuensi kemunculan kata terbanyak setelah dilakukan *text preprocessing*.



Gambar 4.2 Diagram Batang Sepuluh Kata dengan Frekuensi Kemunculan Tertinggi Setelah *Preprocessing*

Beberapa kata-kata yang muncul pada diagram batang frekuensi kemunculan kata tertinggi sebelum dilakukan *text preprocessing* tidak muncul lagi pada diagram batang frekuensi kemunculan kata tertinggi setelah dilakukan *text preprocessing*. Kata "@kai-121" pada Gambar 4.1 merupakan *username* dari akun twitter resmi PT. Kereta Api Indonesia, sehingga kata tersebut tidak muncul pada Gambar 4.2 karena sudah melalui tahapan *cleansing* dimana *username* akan dihapus dari dokumen. Namun, kata "tiket" muncul pada kedua diagram batang hal ini mengindikasikan bahwa kata tersebut paling sering muncul baik sebelum dilakukan *text preprocessing* maupun setelah dilakukan *text preprocessing*. Kata-

kata lain yang terdapat pada Gambar 4.1 merupakan kata-kata yang tidak memiliki makna yang signifikan atau hanya kata penghubung saja, sehingga kata-kata tersebut dihapus pada tahapan *stopword removal*. Pada Gambar 4.2 menunjukkan bahwa sepuluh kata yang paling sering muncul setelah dilakukan *text preprocessing* adalah "tiket", "beli", "jam", "access", "berangkat", "bayar", "info", "lokal", "jadwal", "cek". Sehingga, dari kata-kata yang paling sering muncul tersebut dapat diketahui bahwa sebagian besar *tweet* yang ditujukan kepada PT. Kereta Api Indonesia membahas terkait masalah pembelian tiket dimana salah satu jenis tiketnya adalah tiket lokal. Selain itu juga terdapat pembahasan mengenai info yang dapat berupa info jadwal, jam keberangkatan, pembayaran, maupun tentang KAI access.

Setelah data *tweet* telah dilakukan *text preprocessing* maka tahap selanjutnya adalah melakukan *term weighting* atau konversi data *tweet* menjadi numerik dengan menghitung banyaknya kemunculan kata dasar serta pembobotan dengan menggunakan TF-IDF. Tabel 4.4 merupakan ilustrasi perhitungan frekuensi kemunculan kata dasar.

Tabel 4.3 Ilustrasi Perhitungan *Document Frequency* (DF) dan *Inverse Document Frequency* (IDF)

Tweet	**Kata**				
ke-	**access**	**…**	**tiket**	**…**	**wifi**
1	0	…	0	…	0
2	0	…	1	…	0
⋮	⋮	⋱	⋮	⋱	⋮
2057	0	…	3	…	0
⋮	⋮	⋱	⋮	⋱	⋮
4116	0	…	0	…	0
DF	386	…	1145	…	12
IDF	$\log\left(\frac{4166}{386}\right)$ = 1,0331	…	$\log\left(\frac{4166}{1145}\right)$ = 0,5609	…	$\log\left(\frac{4166}{12}\right)$ = 2,5405

Pada tabel di atas dapat diketahui bahwa *Document Frequency* (DF) kata "access" muncul sebanyak 386 kali dalam seluruh

tweet, kata "tiket" muncul sebanyak 1145 kali, dan kata "wifi" muncul sebanyak 12 kali dari 4166 *tweet*. Perhitungan untuk mendapatkan *Inverse Document Frequency* (IDF) dapat dilihat pada Tabel 4.2 untuk kata "access" mendapatkan IDF sebesar 1,0331, untuk kata "tiket" mendapatkan IDF sebesar 0,5609, dan untuk kata "wifi" mendapatkan nilai 2,5405. Langkah selanjutnya adalah menghitung *Term Frequency – Inverse Document Frequency* (TF-IDF) dengan menggunakan persamaan (2.1) Pada Tabel 4.5 ditampilkan Ilustrasi perhitungan TF-IDF.

Tabel 4.4 Ilustrasi Perhitungan *Term Frequency – Inverse Document Frequency* (TF-IDF)

Tweet	**Kata**				
ke-	**access**	**…**	**tiket**	**…**	**wifi**
1	0	…	0	…	0
2	0	…	0,5609	…	0
⋮	⋮	⋱	⋮	⋱	⋮
2057	0	…	1,6827	…	0
⋮	⋮	⋱	⋮	⋱	⋮
4116	0	…	0	…	0

Tabel 4.5 menunjukkan bahwa dari kata "tiket" pada *tweet* kedua memiliki nilai TF-IDF sebesar 0,5609. Nilai tersebut didapatkan karena TF dari kata "tiket" pada *tweet* ke-2 adalah sebesar satu sehingga diperoleh nilai TF-IDF sebesar $1 \times 0{,}5609 = 0{,}5609$. Begitu juga pada *tweet* ke-2057 yang memiliki TF sebesar 3, maka nilai TF-IDF dari sebesar $3 \times 0{,}5609 = 1{,}6827$. TF-IDF untuk kata-kata yang lain juga dihitung seperti itu. Data pada Tabel 4.5 akan merupakan contoh data unigram atau pemecahan kata menjadi satu kata. Dengan cara perhitungan TF-IDF yang sama juga diterapkan pada data bigram dan trigram. Data yang telah dihitung pembobotnya merupakan data yang akan digunakan untuk langkah selanjutnya yaitu tahap *clustering*

4.2 *Clustering* Menggunakan K-*Means*

Setelah dilakukan *text preprocessing*, maka data seperti Tabel 4.5 diolah lebih lanjut menggunakan *clustering*. Metode *cluste-*

ring yang akan dilakukan pada penelitian ini adalah dengan menggunakan K-*means* dan *Fuzzy* C-*means*. Pada subbab ini akan dibahas *clustering* data menggunakan metode K-*means*. Penentuan jumlah *cluster* (*K*) dihitung mengggunakan Calinski-Harabasz *Index* (CHI) dari *K* sebanyak 2 hingga 20. Penelitian ini membandingkan penggunaan *N-gram*, maka pada metode *clustering* K-*means* ini akan terdapat tiga pilihan *clustering* yaitu K-*means* menggunakan unigram, bigram, dan trigram. Nilai CHI untuk masing-masing *K* dan masing-masing *N-gram* ditampilkan pada Tabel 4.6.

Tabel 4.5 Nilai Calinski-Harabasz *Index* (CHI) dari Metode K-*means*

K	**Nilai Calinski-Harabasz *Index* (CHI)**		
	Unigram	**Bigram**	**Trigram**
2	85,6342	**307,4969**	357,2719
3	**86,7207**	202,1083	**375,4981**
4	77,3089	164,0450	320,5991
5	76,4634	169,9300	321,6601
6	71,2851	145,8467	284,5403
7	67,8643	120,8582	250,2673
8	61,1746	124,9550	249,9494
9	58,9876	124,4258	240,5748
10	57,0922	121,1550	224,9870
11	53,9894	123,1717	234,4873
12	51,5082	115,6494	244,7080
13	49,4404	110,5048	238,7303
14	48,4673	101,8681	233,5641
15	45,8773	111,0804	231,7439
16	45,3998	107,0541	230,9926
17	45,3338	106,7858	233,3480
18	44,3875	103,3862	232,8670
19	42,5523	102,7576	232,5268
20	40,2452	102,1770	232,9092

Berdasarkan Tabel 4.6 dapat diketahui bahwa metode K-*means* dengan *tokenizing* unigram memperoleh jumlah *cluster* optimum sebanyak tiga ($K = 3$) karena nilai evaluasi *cluster* menggunakan Calinski-Harabasz *Index* (CHI) mendapatkan nilai ter-

tinggi yakni 86,7207. *Tokenizing* bigram memiliki nilai CHI tertinggi sebesar 307,4969 pada jumlah *cluster* sebanyak dua ($K = 2$). Sementara itu, untuk metode *clustering* K-*means* dengan *tokenizing* trigram membentuk jumlah *cluster* optimum sebanyak tiga ($K = 3$) karena memperoleh nilai CHI yang paling tinggi yaitu sebesar 375,4981 diantara nilai CHI pada jumlah *cluster* yang lainnya. Sehingga, dapat disimpulkan bahwa jumlah *cluster* optimum dengan metode K-*means* adalah sebanyak tiga *cluster* dengan *tokenizing* trigram. Hal ini dikarenakan nilai CHI yang dihasilkan pada *tokenizing* trigram menghasilkan nilai yang paling tinggi diantara nilai CHI pada *tokenizing* unigram dan bigram.

Setelah didapatkan jumlah *cluster* optimum, maka langkah selanjutnya adalah menentukan *centroid* awal secara acak. Berikut merupakan ilustrasi *centroid* awal.

Tabel 4.6 Ilustrasi *Centroid* Awal

K	**access_beli_ tiket**	**access_cetak _tiket**	**…**	**tiket_via_ access**	**ubah_jadwal _tiket**
1	0,6132	0	…	0	0
2	0	0	…	0,4917	0
3	0	0,7512	…	0	0

Centroid awal yang telah dibentuk seperti pada Tabel 4.7 digunakan untuk menghitung jarak *Euclidean* dari *tweet* ke-*j* ke *centroid* awal pada *cluster* ke-*i*. Pada Tabel 4.8 menampilkan ilustrasi perhitungan jarak *Euclidean* pada *cluster* pertama dengan menggunakan persamaan (2.2).

Tabel 4.7 Ilustrasi Perhitungan Jarak *Euclidean*

Jarak *Euclidean* ($D_{1,j}$)	**Perhitungan Jarak *Euclidean***
$D_{1,1}$	$\sqrt{(0-0,6132)^2+(0-0)^2+\ldots+(0-0)^2+(0-0)^2}=1$
⋮	⋮
$D_{1,1550}$	$\sqrt{(0-0,6132)^2+(0-0)^2+\ldots+(0-0)^2+(0-0)^2}=1,4142$
⋮	⋮

Tabel 4.8 Ilustrasi Perhitungan Jarak Euclidean

Jarak *Euclidean* ($D_{1,j}$)	**Perhitungan Jarak *Euclidean***
$D_{1,2250}$	$\sqrt{(0-0,6132)^2+(0-0)^2+\ldots+(0-0)^2+(0-0)^2}=1,4142$
⋮	⋮
$D_{1,4166}$	$\sqrt{(0-0,6132)^2+(0-0)^2+\ldots+(0-0)^2+(0-0)^2}=1$

Ilustrasi diatas menjelaskan bahwa jarak antara *tweet* ke-2250 ke *centroid* awal pada *cluster* ke-1 adalah sebesar 1,4142. Setelah jarak *Euclidean* dihitung, tahap selanjutnya adalah menentukan pengelompokan data dengan memilih jarak *Euclidean* terkecil yang ditampilkan seperti pada Tabel 4.9.

Tabel 4.9 Ilustrasi Pengelompokan Data

***Tweet* ke-**	**$D_{1,j}$**	**$D_{2,j}$**	**$D_{3,j}$**	**Min($D_{i,j}$)**	**Klaster**
1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮
1550	1,4142	1,4142	0,8246	0,8246	3
⋮	⋮	⋮	⋮	⋮	⋮
2250	1,4142	1,0814	1,4142	1,0814	2
⋮	⋮	⋮	⋮	⋮	⋮
4166	1	1	1	1	1

Dari tabel di atas dapat diketahui bahwa jarak *Euclidean* pada *tweet* ke-1 dan ke-4166 memiliki nilai yang sama pada setiap *cluster*, sehingga *tweet* tersebut dapat dikelompokkan pada salah satu *cluster*. Sedangkan untuk *tweet* ke-1550 merupakan anggota dari *cluster* tiga karena jarak *Euclidean* pada *cluster* tiga memiliki jarak yang paling kecil yaitu sebesar 0,8246. Berbeda halnya dengan *tweet* ke-2250 dimana jarak *Euclidean* terkecil terletak pada *cluster* tiga. Tahap selanjutnya setelah mendapatkan anggota dari setiap *cluster* yaitu membuat *centroid* baru menggunakan persamaan (2.3). *Centroid* baru yang terbentuk digunakan untuk mengukur jarak *Euclidean* dari *tweet* ke *centroid*. Tabel berikut ini menunjukkan ilustrasi *centroid* baru yang terbentuk.

Tabel 4.10 Ilustrasi *Centroid* Baru untuk Iterasi Ke-2

K	access_beli_ tiket	access_cetak _tiket	…	tiket_via_ access	ubah_jadwal _tiket
1	0,0014	0	…	0	0,0014
2	0,0133	0	…	0,2034	0
3	0	0,2457	…	0	0

Tabel 4.10 digunakan untuk menghitung jarak *Euclidean* untuk iterasi ke-2. Jika tidak ada lagi perpindahan objek atau sudah konvergen maka proses iterasi dapat dihentikan.

4.3 *Clustering* Menggunakan *Fuzzy* C-*Means*

Metode *clustering* lain yang dilakukan pada penelitian ini adalah dengan menggunakan *Fuzzy* C-*means*. Penentuan jumlah *cluster* (*K*) pada metode *clustering* ini juga menggunakan nilai Calinski-Harabasz *Index* (CHI). *Clustering* menggunakan *Fuzzy* C-*means* juga dilakukan dengan membandingkan *clustering* menggunakan *N-gram*. Maka, pada metode *clustering* ini juga menghasilkan tiga pilihan *clustering* yaitu *clustering Fuzzy* C-*means* unigram, bigram, dan trigram. Pada penelitian ini, pangkat matriks (*w*) ditetapkan sebanyak 2, maksimum iterasi (MaxIter) sebanyak 1000, dan *error* terkecil yang diharapkan (ξ) sebesar 0,005. Tabel 4.11 menunjukkan nilai CHI pada masing-masing *K* dan masing-masing *N-gram* dari metode *clustering Fuzzy* C-*means*.

Tabel 4.11 Nilai Calinski-Harabasz *Index* (CHI) dari Metode *Fuzzy* C-*means*

K	Nilai Calinski-Harabasz *Index* (CHI)		
	Unigram	**Bigram**	**Trigram**
2	**66,7642**	**193,8511**	240,3925
3	35,4338	192,3724	**254,8247**
4	47,1196	190,3916	251,2408
5	37,5583	189,8484	239,5687
6	38,5560	189,7181	251,2408
7	30,1725	189,7223	240,9742
8	35,5755	189,6458	240,7794
9	24,2775	189,6828	170,8829
10	26,7606	189,6349	170,8829
11	23,9588	189,6349	170,8829

Tabel **4. 12** Nilai Calinski Harabasz *Index* (CHI) dari Metode Fuzzy C-*means* (Lanjutan)

K	**Nilai Calinski-Harabasz *Index* (CHI)**		
	Unigram	**Bigram**	**Trigram**
12	19,3354	189,3587	169,5051
13	24,9671	189,3370	109,6300
14	22,5582	189,3370	169,1042
15	22,3997	189,3203	124,4830
16	25,8144	189,3203	124,0550
17	18,7533	189,3203	109,3239
18	16,2212	189,3128	140,5822
19	16,4769	189,3421	124,2863
20	20,4341	189,3203	169,1042

Pada Tabel 4.11 dan Tabel 4.12 dapat diketahui bahwa metode *clustering Fuzzy* C-*means* dengan *tokenizing* unigram menghasilkan jumlah *cluster* optimum sebanyak 2 *cluster* karena nilai CHI yang dihasilkan memiliki nilai yang paling besar dibanding dengan jumlah *cluster* lain yaitu 66,7642. Untuk *N-gram* sebanyak 2 (bigram) juga menghasilkan jumlah *cluster* optimum sebanyak dua dengan nilai CHI sebesar 193,8511. Sedangkan, untuk *tokenizing* trigram memperoleh jumlah *cluster* optimum sebanyak 3 karena memiliki nilai CHI yang paling besar diantara jumlah *cluster* lainnya yakni sebesar 254,8247. Sehingga, metode *clustering* Fuzzy C-*means* dengan *tokenizing* trigram merupakan metode terbaik yang dihasilkan. Setelah mengetahui jumlah *cluster* yang ingin dibentuk, maka langkah selanjutnya adalah membangkitkan bilangan *random* untuk elemen matriks partisi awal. Tabel 4.13 merupakan ilustrasi matriks partisi awal.

Tabel 4.13 Ilustrasi Matriks Partisi Awal

k	***Tweet* ke-**				
	1	**2**	**…**	**4165**	**4166**
1	0,2913	0,3137	…	0,4116	0,0814
2	0,4354	0,3167	…	0,5008	0,5121
3	0,2732	0,3694	…	0,0874	0,4063

Matriks partisi di atas digunakan untuk menghitung pusat *cluster* ke-*k* dengan menggunakan persamaan (2.4). Persamaan

(4.1) di bawah ini merupakan contoh perhitungan pusat *cluster* pada *cluster* ke-1 untuk kata ke-1 yakni kata "access beli tiket".

$$
\begin{aligned}
V_{1,1} &= \frac{\sum_{i=1}^{n}\left(\left(\mu_{i,1}\right)^{w} x_{i,1}\right)}{\sum_{i=1}^{n}\left(\mu_{i,1}\right)^{w}} \\
&= \frac{\left(\left(\mu_{1,1}\right)^{2} x_{1,1}\right)+\left(\left(\mu_{2,1}\right)^{2} x_{2,1}\right)+\ldots+\left(\left(\mu_{4166,1}\right)^{2} x_{4166,1}\right)}{\left(\mu_{1,1}\right)^{2}+\left(\mu_{2,1}\right)^{2}+\ldots+\left(\mu_{4166,1}\right)^{2}} \\
&= \frac{\left(\left(0,2913\right)^{2}\times 0\right)+\left(\left(0,3137\right)^{2}\times 0\right)+\ldots+\left(\left(0,0814\right)^{2}\times 0\right)}{\left(0,2913\right)^{2}+\left(0,3137\right)^{2}+\ldots+\left(0,0814\right)^{2}} \\
&= 0,0015
\end{aligned}
\tag{4.1}
$$

Pusat *cluster* ke-1 untuk kata pertama yakni "access beli tiket" mendapatkan nilai sebesar 0,0015. Perhitungan seperti pada persamaan (4.1) juga dilakukan pada kata-kata yang lain, sehingga dapat membentuk pusat *cluster* ke-1. Perhitungan serupa juga dilakukan untuk menentukan pusat *cluster* ke-2 dan ke-3, sehingga didapatkan V_{kj} yang dapat ditampilkan pada Tabel 4.14.

Tabel 4.14 Ilustrasi Hasil Perhitungan Pusat *Cluster* ke-*k*

V_{kj}	**Kata**				
	access_beli_ tiket	**access_cetak_ tiket**	**…**	**tiket_via _access**	**ubah_jadwal _tiket**
1	0,0015	0,0005	…	0,0020	0,0013
2	0,0017	0,0016	…	0.0016	0,0019
3	0,0011	0,0011	…	0,0020	0,0010

Pusat *cluster* yang telah didapatkan digunakan untuk menghitung fungsi objektif. Oleh karena itu, setelah mendapatkan pusat *cluster* ke-*k* pada kata ke-*j* (V_{ij}), langkah selanjutnya adalah menghitung fungsi objektif menggunakan persamaan (2.5). Pada persamaan 4.2 merupakan penjabaran perhitungan fungsi objektif dengan pusat *cluster* yang telah dihitung seperti pada Tabel 4.14. Fungsi objektif didapatkan senali 180,1212.

$$P_t = \sum_{i=1}^{n}\sum_{k=1}^{c}\left(\left[\sum_{j=1}^{m}\left(x_{ij}-V_{kj}\right)^2\right]\left(\mu_{ik}\right)^w\right)$$

$$P_1 = \sum_{i=1}^{4166}\sum_{k=1}^{3}\left(\left[\sum_{j=1}^{63}\left(x_{ij}-V_{kj}\right)^2\right]\left(\mu_{ik}\right)^w\right)$$

$$\sum_{i=1}^{4166}\left(\left[\left(x_{i1}-V_{11}\right)^2+\ldots+\left(x_{i,63}-V_{1,63}\right)^2\right]\left(\mu_{i1}\right)^2\right)+ \quad (4.2)$$

$$=\sum_{i=1}^{4166}\left(\left[\left(x_{i1}-V_{21}\right)^2+\ldots+\left(x_{i,63}-V_{2,63}\right)^2\right]\left(\mu_{i2}\right)^2\right)+$$

$$\sum_{i=1}^{4166}\left(\left[\left(x_{i1}-V_{31}\right)^2+\ldots+\left(x_{i,63}-V_{3,63}\right)^2\right]\left(\mu_{i3}\right)^2\right)$$

$$=180{,}1212$$

Kemudian tahapan selanjutnya adalah memperbaiki matriks partisi U berdasarkan persamaan (2.6) sehingga didapatkan derajat keanggotaan baru (matriks partisi) seperti pada Tabel 4.15.

Tabel 4.15 Ilustrasi Hasil Perhitungan Matriks Partisi Baru

V_{ij}	*Tweet* ke-				
	1	**…**	**2052**	**…**	**4166**
1	0,3614	…	0,3332	…	0,3614
2	0,3295	…	0,3331	…	0,3295
3	0,3091	…	0,3337	…	0,3091

Berdasarkan perhitungan fungsi objektif yang telah dihitung seperti pada persamaan (4.2) dapat disimpulkan bahwa $|P_1-P_0|$ = 180,1212 dimana nilai tersebut masih lebih besar dibanding ξ = 0,005. Maka dari itu perhitungan masih terus berlanjut ke iterasi selanjutnya. Demikian seterusnya hinga ($|P_1-P_0|<\xi$) atau $t >$ maksimum iterasi. Setelah proses dapat berhenti, dibentuk matriks partisi akhir. Berdasarkan matriks partisi akhir yang telah terbentuk dapat diperoleh informasi mengenai kecenderungan *tweet* tersebut dikelompokkan pada *cluster* yang mana. Suatu *tweet* memiliki derajat keanggotaan tertentu untuk menjadi anggota kelompok. Derajat

keanggotaan terbesar menunjukkan kecenderungan tertinggi suatu *tweet* untuk masuk menjadi anggota kelompok. Tabel 4.16 menunjukkan derajat keanggotaan tiap *tweet* pada setiap *cluster* beserta kecenderungan tertinggi suatu *tweet* untuk masuk dalam suatu kelompok.

Tabel 4.16 Derajat Keanggotaan dan Penentuan *Cluster* Tiap *Tweet*

***Tweet* ke-**	**Derajat Keanggotaan (μ_i) pada Cluster ke-**			**Max(μ_{ik})**	***Cluster***
	1	**2**	**3**		
1	0,9997	0,0001	0,0001	0,9997	1
⋮	⋮	⋮	⋮	⋮	⋮
1550	0,3279	0,3360	0,3360	0,3360	2
⋮	⋮	⋮	⋮	⋮	⋮
2250	0,3136	0,3432	0,3431	0,3431	3
⋮	⋮	⋮	⋮	⋮	⋮
4166	0,9997	0,0001	0,0001	0,9997	1

4.4 Perbandingan Antar Metode *Clustering*

Setelah melakukan *clustering* menggunakan metode K-*means* maupun Fuzzy C-*means* maka dilakukan perbandingan antar metode *clustering* berdasarkan nilai CHI yang tertinggi guna mendapatkan metode *clustering* terbaik yang digunakan untuk mengelompokkan *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia. Tabel 4.17 merupakan perbandingan metode *clustering* K-*means* dan Fuzzy C-*means* dengan jumlah *cluster* optimum pada masing-masing metode.

Tabel 4.17 Perbandingan Antar Metode *Clustering*

Metode *Clustering*	***N-gram***	***K***	**CHI**
K-*means*	**Trigram**	**3**	**375,4981**
Fuzzy C-*means*	Trigram	3	254,8247

Seperti terlihat pada Tabel 4.17 tampak metode *clustering* yang menghasilkan nilai CHI tertinggi adalah sebesar 375,4981 yang mana nilai tersebut dihasilkan apabila menggunakan metode *clustering* K-*means* dengan *tokenizing* trigram. Sehingga dapat disimpulkan bahwa metode tersebut merupakan metode *clustering*

terbaik untuk mengelompokkan *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia. Jumlah anggota pada tiap *cluster* yang terbentuk dapat dilihat pada Tabel 4.18.

Tabel 4.18 Jumlah Anggota Tiap *Cluster*

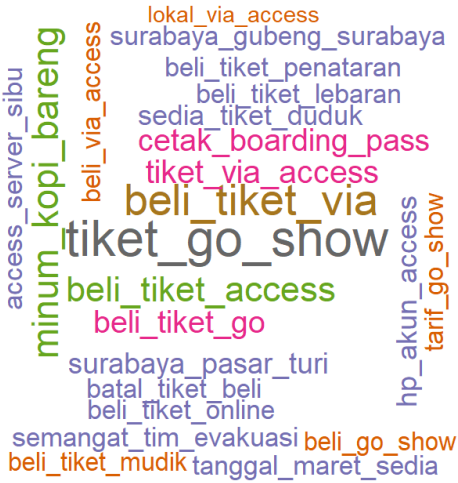
***Cluster* ke-**	**Jumlah Anggota**
1	4070
2	58
3	38

Seperti yang dapat dilihat pada Tabel 4.18 dapat diketahui bahwa *cluster* yang memiliki anggota paling banyak adalah *cluster* ke-1 dengan jumlah anggota sebanyak 4070 *tweet*. Lalu diikuti dengan *cluster* ke-2 dengan anggota sebanyak 58 *tweet*. Sedangkan, *cluster* ke-3 merupakan *cluster* dengan jumlah anggota paling sedikit yaitu sebanyak 38 *tweet*. Hal ini menunjukkan bahwa *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia sebagian besar membahas mengenai hal yang serupa. Untuk mendeskripsikan pembahasan pada setiap *cluster* maka dilakukan visualisasi dari tiga *cluster* yang telah terbentuk menggunakan *word cloud*.

4.5 Visualisasi Menggunakan *Word cloud*

Pada subbab ini akan dilakukan visualisasi menggunakan *word cloud* pada setiap *cluster* yang telah terbentuk berdasarkan hasil *clustering* terbaik yang telah didapatkan pada subbab sebelumnya, yaitu metode *clustering* K-*means* dengan menggunakan trigram dimana *cluster* yang terbentuk adalah sebanyak tiga *cluster*. Tujuan dibuatnya visualisasi pada setiap *cluster* adalah untuk mengetahui kategori *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia (@KAI121) selain itu penggunaan *word cloud* juga jauh lebih menarik dibanding dengan diagram frekuensi. Tiap *cluster* divisualisasikan dengan menggunakan *word cloud* dimana ukuran *font* setiap kata menandakan frekuensi kemunculan dari kata yang bersangkutan sehingga semakin besar ukuran *font* maka semakin banyak pula frekuensi kemunculan dari kata tersebut. Berikut merupakan *word cloud* dari setiap *cluster*

yang telah terbentuk. Pada Gambar 4.3 menunjukkan *word cloud* dari *cluster* satu.



Gambar 4.3 *Word Cloud Cluster* Satu

Gambar 4.3 menunjukkan *word cloud* dari *cluster* 1. Dilihat dari *word cloud* tersebut dapat diketahui bahwa kata-kata yang paling sering muncul adalah "tiket *go-show*", lalu diikuti dengan "beli tiket via". Hal ini menunjukkan bahwa pada *cluster* 1 pembahasan yang banyak dibincangkan oleh pelanggan adalah tentang permasalahan tiket *go-show*. Tiket *go-show* merupakan pembelian tiket mendadak pada hari keberangkatan. Berikut merupakan sepuluh kata-kata yang paling sering muncul pada *cluster* satu.

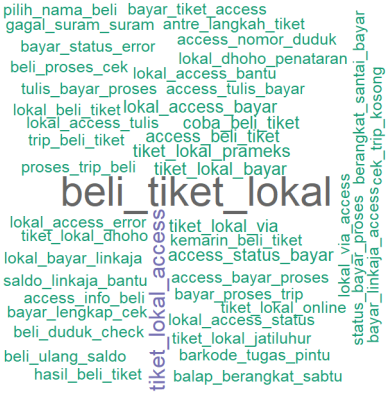
Tabel 4.19 Frekuensi Kemunculan Kata Tertinggi Pada *Cluster* Satu

No.	Kata-Kata	Frekuensi
1	tiket_go_show	27
2	beli_tiket_via	22
3	minum_kopi_bareng	15
4	beli_tiket_access	14
5	tiket_via_access	12
6	beli_tiket_go	11
7	cetak_boarding_pass	11

Tabel 4.20 Frekuensi Kemunculan Kata Tertinggi Pada *Cluster* Satu (lanjutan)

No.	Kata-Kata	Frekuensi
8	jam_beli_tiket	11
9	info_beli_tiket	10
10	surabaya_pasar_turi	9

Berdasarkan tabel di atas dapat diketahui bahwa permasalahan yang sering disampaikan pelanggan ke akun Twitter resmi PT. Kereta Api Indonesia adalah pembelian tiket *go-show*. Kategori pada *cluster* ini disampaikan dengan pertanyaan atau keluhan tentang info pembelian tiket yang berupa pencetakan *boarding pass* ataupun jam keberangkatan. Selain itu juga terdapat kata-kata "minum kopi bareng", sehingga dapat diperoleh informasi juga bahwa penumpang KAI banyak membincangkan kegiatan yang diadakan oleh KAI ini. Kegiatan ini bertujuan untuk mengenalkan kopi lokal Indonesia kepada penumpang kereta api dengan cara menyuguhkan kopi secara gratis di beberapa stasiun maupun di atas kereta api. Syarat agar penumpang mendapatkan kopi gratis ini adalah dengan menunjukkan aplikasi KAI access yang sudah terdaftar di ponsel kepada petugas. Pada *cluster* ini banyak ditemukan *tweet* yang membahas tentang kereta api jarak jauh yang dibeli pada saat hari keberangkatan (*go-show*). Kereta api yang ditanyakan antara lain kereta Wijayakusuma dengan rute Cilacap-Banyuwangi, kereta api Pasundan dengan rute Kiaracondong-Surabaya Gubeng, lalu juga ada kereta api Kahuripan dengan rute Blitar-Kiaracondong, kereta api tawang jaya dengan rute Semarang-Jakarta Pasar Senen dan lain kereta api lainnya. Hal ini menandakan bahwa tiket kereta api jarak jauh banyak dibincangkan di Twitter oleh masyarakat dengan pembelian pada hari keberangkatan. Selain itu pada *cluster* satu ini juga terdapat *tweet* yang membahas tentang kritikan maupun sarana tentang fasilitas atau sarana prasarana yang disediakan seperti toilet, kursi, dan lainnya. Sehingga, pada *cluster* satu ini dapat dijadikan bahan peningkatan kualitas pelayanan yang disediakan PT. Kereta Api Indonesia terutama pelayanan fasilitas sarana dan prasarana. Selanjutnya *word cloud* pada *cluster* dua dapat ditunjukkan pada Gambar 4.4.



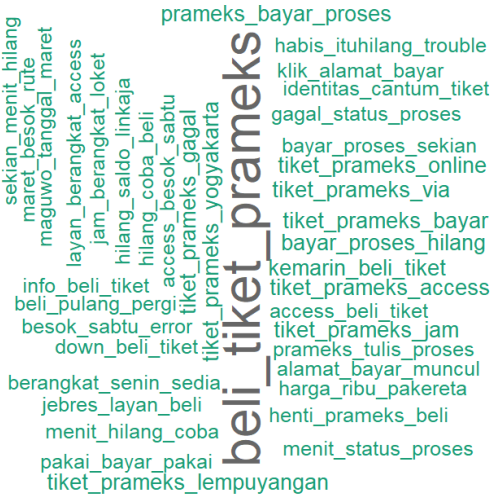
Gambar 4.4 *Word Cloud Cluster* Dua

Tampak pada Gambar 4.4 kata-kata yang memiliki ukuran *font* paling besar merupakan "beli tiket lokal" dimana ukuran tersebut sangat besar dibanding dengan kata-kata disekitarnya. Hal ini menandakan bahwa dari 58 *tweet* yang terdapat pada *cluster* 2 sebagian besar membahas tentang pembelian tiket lokal. Kata-kata setelahnya yang memiliki ukuran *font* yang cukup besar yakni "tiket lokal access" yang artinya pembelian tiket lokal dilakukan pada via aplikasi yang bernama KAI Access. Tabel berikut menampilkan kata-kata yang paling sering muncul pada *cluster* dua.

Tabel 4.21 Frekuensi Kemunculan Kata Tertinggi Pada *Cluster* Dua

No.	Kata-Kata	Frekuensi
1	beli_tiket_lokal_	58
2	tiket_lokal_access	15
3	access_beli_tiket	6
4	tiket_lokal_bayar	6
5	tiket_lokal_prameks	5
6	tiket_lokal_via	4
7	kemarin_beli_tiket	3
8	lokal_access_bayar	3
9	lokal_beli_tiket	3
10	tiket_lokal_beli	3

Dilihat dari tabel frekuensi kemunculan kata tertinggi pada *cluster* dua dapat diketahui bahwa *cluster* dua merupakan kategori *tweet* yang membahas tentang permasalahan pembelian tiket lokal dimana permasalahan yang banyak terjadi adalah ketika menggunakan aplikasi KAI Access. Permasalahan yang muncul adalah tentang kegagalan proses pembayaran pembelian tiket kereta api lokal menggunakan Link Aja yang mana calon penumpang tidak dapat membeli tiket kereta api lokal karena status pembayaran tetap tertera masih diproses sedangkan saldo Link Aja masih tersedia. Selain itu, pembelian tiket lokal dengan KAI Access juga menimbulkan permasalahan tentang pemilihan kursi atau tempat duduk. Anggota pada *cluster* ini berjumlah 58 *tweet* dimana 21 diantaranya dibincangkan pada tanggal 13 Maret 2019 dan mengeluhkan pemesanan tiket kereta api lokal melalui KAI Access yang mengalami kegagalan proses pembayaran.



Gambar 4.5 *Word Cloud Cluster* Tiga

Kata-kata yang memiliki ukuran *font* paling besar pada *Word cloud cluster* 3 adalah “beli tiket prameks” sehingga dapat

diketahui bahwa pembahasan yang sering muncul berhubungan dengan pembelian tiket prameks. Kereta api prameks (prambanan ekspres) beroperasi dalam bentuk komuter ekonomi yang menghubungkan Kutoarjo, Yogyakarta, dan Solo Balapan. Tabel 4.20 menunjukkan frekuensi kemunculan kata tertinggi pada *cluster* tiga.

Tabel 4.22 Frekuensi Kemunculan Kata Tertinggi Pada Cluster Tiga

No.	Kata-Kata	Frekuensi
1	beli_tiket_prameks	38
2	beli_beli_tiket	4
3	kemarin_beli_tiket	3
4	tiket_prameks_jam	3
5	tiket_prameks_access	3
6	tiket_prameks_yogyakarta	3
7	prameks_jam_beli	2
8	prameks_bayar_proses	2
9	tiket_prameks_bayar	2
10	tiket_prameks_lempuyangan	2

Permasalahan yang muncul untuk pembelian tiket prameks juga sama dengan pembelian tiket lokal dimana status pembayaran masih diproses sedangkan saldo Link Aja sudah terpotong, namun beberapa menit kemudian pesanan hilang. Selain itu juga terdapat beberapa penumpang yang menanyakan perihal pembelian tiket untuk orang lain sedangkan nama yang tercantum pada tiket bukanlah nama calon penumpang. Pada *cluster* tiga ini juga banyak dibincangkan pada tanggal 13 maret 2019 dimana 14 *tweet* dari 38 *tweet* yang berada di *cluster* tiga membincangkan tentang tiket prameks. Sehingga dapat dibuat ringkasan dari setiap *cluster* seperti pada Tabel 4.23 berikut ini.

Tabel 4.23 Anggota *Cluster* dan Keterangan Tiap *Cluster*

Cluster	**Anggota *Cluster***	**Keterangan**
1	4070	Pembelian Tiket *go-show*
2	58	Pembelian Tiket Lokal
3	38	Pembelian Tiket Prameks

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis dan pembahasan yang telah dilakukan pada bab 4, maka diperoleh kesimpulan sebagai berikut.

1. Kata yang paling banyak muncul sebelum dilakukan *text preprocessing* dari 4166 *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia (@KAI121) periode 8 Maret 2019 hingga 18 Maret 2019 adalah "@KAI121", "di", "min", "kereta", "tiket", "saya", "bisa", "ada", "untuk", dan "mau". Sehingga perlu adanya *text preprocessing* untuk menghilangkan kata-kata yang tidak memiliki makna. Pada penelitian *text preprocessing* yang dilakukan yaitu *cleansing*, *case folding*, *stemming*, normalisasi kata, *stopwords removal*, dan *tokenizing*. Setelah dilakukan *text preprocessing* kata-kata yang paling banyak muncul berubah menjadi "tiket", "beli", "jam", "access", "berangkat", "bayar", "info", "lokal", "jadwal", "cek".
2. Metode K-*means* dengan *N-gram* sebanyak 3 (trigram) merupakan metode terbaik untuk mengelompokkan *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia. Hal ini dikarenakan metode tersebut menghasilkan nilai CHI yang paling besar dibandingkan dengan metode Fuzzy C-*means* yaitu sebesar 375,4981. *Clustering* dengan metode terbaik memperoleh jumlah *cluster* optimum sebanyak tiga. Sehingga terdapat tiga kategori *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia (@KAI121).
3. Dari *word cloud* setiap *cluster* yang telah dibuat, dapat diketahui bahwa *tweet* yang ditujukan kepada akun Twitter resmi PT. Kereta Api Indonesia terbagi menjadi tiga kategori *tweet*. Tiga kategori tersebut adalah tentang pembelian tiket *go-show* dimana tiket ini dibeli saat hari keberangkatan, pembelian tiket lokal yang mana tiket ini beroperasi dalam satu provinsi, dan

pembelian tiket prameks (prambanan ekspress) dimana rute perjalanannya adalah Yogyakarta, Solo, dan Kutoarjo.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, maka saran yang dapat diberikan kepada PT. Kereta Api Indonesia adalah memisahkan *tweet* yang masuk berdasarkan kategori, menanggapi setiap *tweet* yang masuk dengan membuat *template* jawaban untuk setiap kategori agar dapat mempercepat dan mempermudah. Untuk penelitian selanjutnya, dapat dikembangkan dengan meningkatkan *preprocessing* agar hasil yang diperoleh dapat lebih baik. Selain itu, kata "prameks" juga dapat di *similarity* menjadi "lokal" karena kereta api prameks juga merupakan kereta api lokal.

DAFTAR PUSTAKA

Adriani, M., Asian, J., Nazief, B., Tahaghoghi, S.M.M., Williams, H.E. 2007. Stemming Indonesian: A Confi x-Stripping Approach. *ACM Transaction on Asian Language Information Processing (TALIP)*, 6(4), pp. 1-33.

Badan Pusat Statistik, 2018. Jumlah Penumpang Kereta Api 2006-2018. [Online] Available at: https://www.bps.go.id/linkTableDinamis/view/id/815 [Diakses 16 Februari 2019].

Caliński, T., & Harabasz, J. (1974). A Dendrite Method for Cluster Analysis. *Communications in Statistics*, *3*(1), 1-27.

Chasbullah, S., 2018. *Analisis Berita Hoax Menggunakan Klasifikasi Multinomial Naive Bayes dengan Fitur N-gram*, Bogor: Institut Pertanian Bogor.

Databooks, 2018. KataData. [Online] Available at: https://databoks.katadata.co.id/datapublish [Diakses 14 Februari 2019].

Feldman, R. & Sanger, J., 2007. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York: Cambridge University Press.

Han, J., Kamber, M. & Pei, J., 2012. *Data Mining: Concepts and Techniques*. 3rd penyunt. USA: Elsevier Inc.

Harjanta, A. T. J., 2015. Preprocessing Text Meminimalisir Kata yang Tidak Berarti dalam Proses Text Mining. *Jurnal Informatika UPGRIS*, Volume I, pp. 1-9.

Hayatin, N., Fatichah, C. & Purwitasari, D., 2015. Pembobotan Kalimat Berdasarkan Fitur Berita dan Trending Issue untuk Peringkasan Multi Dokumen Berita. *Jurnal Ilmiah Teknologi Informasi*, 13(1), pp. 38-44.

Isnan, M. A. & Sutopo, 2014. Analisis Pengaruh Kualitas Pelayanan Reservation Ticket Terhadap Kepuasan Pelanggan. *Jurnal Studi Manajemen & Organisasi*, 11(2), pp. 143-152

Jannah, S. Z., Fithriasari, K., Prastyo, D. D. & Iriawan, N., 2018. Text Mining for Identifying and Visualizing Topics of Citizen Opinion in. *International Conference on Theoretical and Applied Statistics*, p. 82

Johnson, R. A. & Wichern, D. W., 2007. *Applied Multivariate Statistical Analysis*. 6th penyunt. United States of America: Pearson Prentice Hall.

Karyadi, S., Yasin, H. & Mukid, M. A., 2016. Analisis Kecenderungan Informasi dengan Menggunakan Metode Text Mining. *Jurnal Gaussian*, 5(4), pp. 763-770.

Kusumadewi, S. & Prunomo, H., 2010. *Aplikasi Logika Fuzzy untuk Pendukung Keputusan*. 2nd *penyunt*. Yogyakarta: Graha Ilmu.

Majumder, P., Mitra, M. & Chauduri, B. B., 2002. *N-gram: A Language Independent Approach to IR and NLP*. ICUKL.

Manning, C. D., Raghavan, P. & Schutze, H., 2009. *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

McNaught, C. & Lam, P., 2010. Using Wordle as a Supplementary Research Tool. *The Qualitative Report*, 15(3), pp. 630-643.

Patil, D. B. & Dongre, Y. V., 2015. A Fuzzy Approach for Text Mining. *International Journal Mathematical Sciences and Computing*, Issue 4, pp. 34-43.

PT. Kereta Api Indonesia, 2016. *Company Profile PT Kereta Api Indonesia (Persero).*

Riyadi, M. A., Pratiwi, D. S., Irawan, A. R., & Fithriasari, K. (2017). Clustering stationary and non-stationary time series based on autocorrelation distance of hierarchical and k-means algorithms. *International Journal of Advancesin in Intelligent Informatics*, *3*(3), 154-160.

Talib, R., Hanif, M. K., Ayesha, S. & Fatima, F., 2016. Text Mining : Techniques, Applications and Issues. *International*

Journal of Advanced Computer Science and Applications, 7(11), pp. 414-418.

Weiguo, F., Wallace, L. & Rich, S., 2006. Tapping The Power of Text Mining. *Communications of the ACM*, 49(9), pp. 77-82.

Wijaya, A. & Sensuse, D. I., 2011. *Perencanaan Strategis Sistem Informasi dan Teknologi Informasi Pada Perusahaan Otomotif dengan Menggunakan Metodologi Tozer*. Yogyakarta, Seminar Nasional Aplikasi Teknologi Informasi.

Witten, I. H. & Frank, E., 2012. *Data Mining Practical Machine Learning Tools and Techniques.* 2nd *penyunt*. San Francisco: Morgan Kaufmann.

Yan, J., Michael & Power, J., 1994. *Using Fuzzy Logic (Toward Intelligent System.* New York: Prentice-Hall.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

Lampiran 1. Data *Tweet*

	text
1	terimakasih @KAI121 , pembayaran melalui @linkaja sudah bisa di aplikasi berbasis iOS. Mantap betul dan cepat! Yang android kapan min? https://t.co/d22TlPATdj
2	@qonitalyn Waktu itu saya naik di cek ktp nya :) mungkin yang jaga berbeda . @KAI121 meski Ka lokal tetap dicek ya ktp sama e-tiketnya ? Terima kasih
3	Apakah ada rekruitmen pt kai di bulan maret ini? Terima kasih @KAI121
4	@KAI121 Kenapa nggak diperpanjang saja KA Jayakarta premium sampai Malang?
5	Min kalau saya dari stasiun tulungagung mau ke stasiun sidoarjo, itu naik kereta apa turun di mana? Saya cek gak ada rute itu untuk 12 maret. bisa di pesan H- satu minggu kan? @KAI121
⋮	⋮
4356	"pengen naik MRT. Kayaknya canggih ya bisa naik kereta dalam tanah. Ternyata indonesia bisa juga buat kereta dalam tanah, ya walaupun baru beberapa kilometer doang. tapi, itu adalah awal yang bagus dan tidak terlalu terlambat. min, PT @KAI121 gak pengen bikin MRT juga?"
4357	Min @KAI121 untuk penukaran 15 boarding pass apakah bisa dilakukan di stasiun Kediri? Trims.
4358	@KAI121 pagi min, kereta penataran dhoho dari blitar ke malang untuk besok jam 4.45 masih ada tempat duduk?
4359	@KAI121 Metode pembayaran hanya bisa menggunakan LinkAja ya min?
4360	@realaryo @KAI121 Harusnya ini tiba pukul 06.06 di Gambir. Tapi ini sekarang baru melewati stasiun bekasi.

Lampiran 2. *Syntax Crawling* Data Menggunakan RStudio

```
# Load packages
library(rtweet)
library(tidyverse)

# Twitter authentication
twitter_token <- create_token(
  app = "app name",
  consumer_key = 'your consumer key',
  consumer_secret = 'your consumer secret',
  access_token = 'your access token',
  access_secret = 'your access secret')

# Retrieve tweets
tweetKAI <- search_tweets("@KAI121", n = 30000,
                    include_rts = FALSE, token = twitter_token)
                      %>%
                    select(status_id, created_at, screen_name, text)

# Short Cleaning
tweetKAI$created_at2 <- tweetKAI$created_at %>%
  str_sub(1,10)
tweetKAI$screen_name2 <- tweetKAI$screen_name %>%
  str_to_lower
tweetKAI <- tweetKAI[order(tweetKAI$created_at2),] %>%
  subset(!duplicated(tweetKAI$status_id))

# Export Data
write.csv(tweetKAI, file = "20190318_tweet_KAI.csv")
```

Lampiran 3. *Syntax Text Preprocessing* dan Karakteristik Data menggunakan Python

```
import pandas as pd
import re
import nltk
from Sastrawi.Stemmer.StemmerFactory import
StemmerFactory
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

#Import Data
data = pd.read_csv(r20190318_tweet_KAI.csv',header=0,
            encoding='latin-1')
text = data['text']

#Menghapus Link
datanolink = []
for line in text:
    result = re.sub(r"http\S+", " ", line)
    datanolink.append(result)

# Menghapus Simbol Retweet
datanort = []
for line in datanolink:
    result = re.sub(r"RT", " ", line)
    datanort.append(result)

# Menghapus Username
datanousername = []
for line in datanort:
    result = re.sub(r"@\S+", " ", line)
    datanousername.append(result)
```

```
#Menghapus baris baru
datanoline=[]
for line in datanousername :
    result=re.sub("\n"," ",line)
    datanoline.append(result)

#Menghapus angka
datanonum=[]
for line in datanoline :
    result=re.sub("\d"," ",line)
    datanonum.append(result)

#Menghapus hastag
datanohastag=[]
for line in datanonum :
    result=re.sub(r"#\S+","",line)
    datanohastag.append(result)

#Menghapus emoticon
datanoemoticon=[]
for line in datanohastag :
    result = re.sub(r'<.*?>',"",line)
    datanoemoticon.append(result)

#Menghapus tanda baca
datanopunctuation=[]
for line in datanoemoticon :
    result=re.sub(r"[^\w\s]"," ",line)
    datanopunctuation.append(result)
```

```
#Menghapus spasi berlebih
datanodoublespace=[]
for line in datanopunctuation :
    result=re.sub(r'\s+',' ',line)
    datanodoublespace.append(result)
cleansing = pd.DataFrame(datanodoublespace,
columns=['Cleansing'])

pd.set_option('display.max_colwidth', 0)
tweet = pd.DataFrame(data['text'])
tweet['Cleansing']=cleansing
tweet.head()

#Case Folding
data_lower = []
for line in datanodoublespace:
    a = line.lower()
    data_lower.append(a)
lower = pd.DataFrame(data_lower, columns=['case folding'])
tweet['Case Folding']=lower
tweet.head()

#Stemming
factory = StemmerFactory()
stemmer = factory.create_stemmer()
data_stemmed = map(lambda x: stemmer.stem(x), data_lower)
datastemmed = list(data_stemmed)
stem=pd.DataFrame(datastemmed, columns=['Stemming'])
tweet['Stemming']=stem
tweet.head()
```

```
#sinonim
kata = {" pesan":" beli", "pesan ":"beli ", "bas ":"basis ",
"informasi ":"info ", "bales ":"balas ", "toreh ":"menoreh
",…."mjk ":"mojokerto ", "disimpenin ":"simpan ", "type
":"tipe ", "ketidaknyamannya ":"tidak nyaman ", "makasih
":"terima kasih "}

def replace_all(text, dic):
        for i, j in dic.items():
            text = text.replace(i, j)
        return text

import collections
from collections import OrderedDict
dic = OrderedDict(kata)

datachange = []
for line in datastemmed:
    result = replace_all(line, dic)
    datachange.append(result)
replace = pd.DataFrame(datachange, columns=['Normalisasi'])
tweet['Normalisasi Kata']=replace
tweet.head()

#Menghapus stopword
stopword_list = open(r'stopwords.txt',encoding='latin
     1').read()
datafinal =[]

for line in datachange:
    word_token = nltk.word_tokenize(line)
    word_token = [word for word in word_token if not word in
                stopword_list and not word[0].isdigit()]
    datafinal.append(" ".join(word_token))
```

```
stopword = pd.DataFrame(datafinal, columns=['stopword'])
tweet['Stopword']=stopword
tweet.head()

databersih = pd.DataFrame (tweet['Stopword'])
databersih.to_csv(r'databersih.csv')

#DTM UNIGRAM
vectorizer = CountVectorizer(min_df=10)
DTM = vectorizer.fit_transform(datafinal)
DTM_ = pd.DataFrame(DTM.toarray(),
                columns = vectorizer.get_feature_names())
DTM_.to_csv(r'UNIGRAM_DTM.csv')

#TF-IDF UNIGRAM
vectorizer = TfidfVectorizer(min_df=10)
TFIDF = vectorizer.fit_transform(datafinal)
TFIDF_ = pd.DataFrame(TFIDF.toarray(),
                  columns = vectorizer.get_feature_names())
TFIDF_.to_csv(r'UNIGRAM_TFIDF.csv')

#DTM BIGRAM
vectorizer = CountVectorizer(min_df=10, ngram_range=(2,2))
DTM_BG = vectorizer.fit_transform(datafinal)
term = vectorizer.get_feature_names()
term1 = [word.replace(' ','_') for word in term]
DTM_BG_ = pd.DataFrame(DTM_BG.toarray(),
                       columns = term1)
DTM_BG_.to_csv(r'BIGRAM_DTM.csv')
```

```
#TF-IDF BIGRAM
vectorizer = TfidfVectorizer(min_df=10, ngram_range=(2,2))
TFIDF_BG = vectorizer.fit_transform(datafinal)
term = vectorizer.get_feature_names()
term1 = [word.replace(' ','_') for word in term]
TFIDF_BG_ = pd.DataFrame(TFIDF_BG.toarray(),
                         columns = term1)
TFIDF_BG_.to_csv(r'BIGRAM_TFIDF.csv')

#DTM TRIGRAM
vectorizer = CountVectorizer(min_df=5, ngram_range=(3,3))
DTM_TG = vectorizer.fit_transform(datafinal)
term = vectorizer.get_feature_names()
term1 = [word.replace(' ','_') for word in term]
DTM_TG_ = pd.DataFrame(DTM_TG.toarray(),
                       columns = term1)
DTM_TG_.to_csv(r'TRIGRAM_DTM.csv')

#TF-IDF TRIGRAM
vectorizer = TfidfVectorizer(min_df=5, ngram_range=(3,3))
TFIDF_TG = vectorizer.fit_transform(datafinal)
term = vectorizer.get_feature_names()
term1 = [word.replace(' ','_') for word in term]
TFIDF_TG_ = pd.DataFrame(TFIDF_TG.toarray(),
                         columns = term1)
TFIDF_TG_.to_csv(r'TRIGRAM_TFIDF.csv')

# Karakteristik Data
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from collections import Counter
```

```
mpl.rcParams['figure.figsize']=(10.0,8.0)
mpl.rcParams['font.size']=12
mpl.rcParams['savefig.dpi']=100
mpl.rcParams['figure.subplot.bottom']=.1

#Sebelum preprocessing
kata = [s.lower().split() for s in text if s]
noline_ = [sublist for l in kata for sublist in l]

counts1 = dict(Counter(noline_).most_common(10))
labels1, values1 = zip(*counts1.items())
# sort values in descending order
indSort1 = np.argsort(values1)[::-1]

# rearrange data
labels1 = np.array(labels1)[indSort1]
values1 = np.array(values1)[indSort1]
indexes1 = np.arange(len(labels1))
mybar=plt.bar(indexes1, values1)
# get rid of the frame
for spine in plt.gca().spines.values():
    spine.set_visible(False)
# remove all the ticks and directly label each bar with
respective value
plt.tick_params(top='off', bottom='off', left='off', right='off',
labelleft='off', labelbottom='on')
# direct label each bar with Y axis values
for bari in mybar:
    height = bari.get_height()
    plt.gca().text(bari.get_x() + bari.get_width()/2,
bari.get_height() + 100, str(int(height)), ha='center',
                color='black', fontsize=15)
```

```
# add labels
plt.xticks(indexes1, labels1)
plt.show()

#Setelah Preprocessing
after = [s.lower().split() for s in datafinal if s]
after_ = [sublist for l in after for sublist in l]

counts2 = dict(Counter(after_).most_common(10))
labels2, values2 = zip(*counts2.items())
# sort values in descending order
indSort2 = np.argsort(values2)[::-1]

# rearrange data
labels2 = np.array(labels2)[indSort2]
values2 = np.array(values2)[indSort2]
indexes2 = np.arange(len(labels2))
mybar2=plt.bar(indexes2, values2)
# get rid of the frame
for spine in plt.gca().spines.values():
    spine.set_visible(False)
# remove all the ticks and directly label each bar with
respective value
plt.tick_params(top='off', bottom='off', left='off', right='off',
labelleft='off', labelbottom='on')
# direct label each bar with Y axis values
for bari in mybar2:
    height = bari.get_height()
    plt.gca().text(bari.get_x() + bari.get_width()/2,
bari.get_height() + 25, str(int(height)), ha='center',
                color='black', fontsize=15)
# add labels
plt.xticks(indexes2, labels2)
plt.show()
```

Lampiran 4. *Syntax Text Clustering* Menggunakan Python

```
# K-Means
# optimal number of cluster by using CHI (unigram)
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
import numpy as np
from sklearn.cluster import KMeans
chi = {}
for k in range(2, 21):
   kmeans_model =
KMeans(n_clusters=k,random_state=1).fit(TFIDF_)
   labels = kmeans_model.labels_
   chi[k] = metrics.calinski_harabaz_score(TFIDF_,labels)
   print("For n_clusters={}, The CHI is {}".format(k, chi[k]))

# optimal number of cluster by using CHI (bigram)
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
import numpy as np
from sklearn.cluster import KMeans
chi = {}
for k in range(2, 21):
   kmeans_model =
KMeans(n_clusters=k,random_state=1).fit(TFIDF_BG_)
   labels = kmeans_model.labels_
   chi[k] = metrics.calinski_harabaz_score(TFIDF_BG_,labels)
   print("For n_clusters={}, The CHI is {}".format(k, chi[k]))
```

```
# optimal number of cluster by using CHI (trigram)
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
import numpy as np
from sklearn.cluster import KMeans
chi = {}
for k in range(2, 21):
   kmeans_model =
KMeans(n_clusters=k,random_state=1).fit(TFIDF_TG_)
   labels = kmeans_model.labels_
   chi[k] = metrics.calinski_harabaz_score(TFIDF_TG_,labels)
   print("For n_clusters={}, The CHI is {}".format(k, chi[k]))

#K-Means for optimal number of clusters (K=3 TRIGRAM)
from sklearn.cluster import KMeans
num_clustersTG = 3
km_TG =
KMeans(n_clusters=num_clustersTG,random_state=1)
%time km_TG.fit(TFIDF_TG_)
clustersTG = km_TG.labels_.tolist()
y_kmeansTG = km_TG.predict(TFIDF_TG_)
data['klaster'] = pd.Series(y_kmeansTG, index=data.index)

#size of each cluster TRIGRAM
from sklearn.externals import joblib
joblib.dump(km_TG, 'doc_cluster.pkl')
km_TG = joblib.load('doc_cluster.pkl')
clustersTG = km_TG.labels_.tolist()
clustersTG_df=pd.DataFrame(clustersTG)
clustersTG_df['klaster']=pd.DataFrame(clustersTG)
clustersTG_df['klaster'].value_counts()
```

```
#data frame of clustering result TRIGRAM
resultTG_df=pd.DataFrame(datafinal)
resultTG_df.columns=['text']
resultTG_df['klaster']=pd.Series(y_kmeansTG,index=data.inde
x)
resultTG_df.to_csv(r'E:\00. Kuliah\SEMESTER 8\TUGAS
AKHIR\LAPORAN\OLAHAN FIX\CLUSTER KMEANS
TRIGRAM 3.csv')
pd.DataFrame(km_TG.cluster_centers_).to_csv(r'E:\00.
Kuliah\SEMESTER 8\TUGAS AKHIR\LAPORAN\OLAHAN
FIX\Centroid Kmeans Trigram.csv')

#DTM TRIGRAM MIN_DF=0 (UNTUK VISUALISASI
WORDCLOUD)
vectorizer = CountVectorizer(min_df=0, ngram_range=(3,3))
DTM_TG = vectorizer.fit_transform(datafinal)
term = vectorizer.get_feature_names()
term1 = [word.replace(' ','_') for word in term]
DTM_TG_ = pd.DataFrame(DTM_TG.toarray(), columns =
term1)
DTM_TG_.to_csv(r'E:\00. Kuliah\SEMESTER 8\TUGAS
AKHIR\LAPORAN\OLAHAN
FIX\MINDF0_TRIGRAM_DTM.csv')


# Fuzzy C-Means
import skfuzzy
from sklearn.metrics import silhouette_score
from sklearn import metrics
import numpy as np

TFIDF_fuzz = TFIDF_.transpose()
TFIDFBG_fuzz = TFIDF_BG_.transpose()
TFIDFTG_fuzz = TFIDF_TG_.transpose()
```

```
# Unigram
cluster_fuzz = pd.DataFrame()
cntr_2, u_2, u0_2, d_2, jm_2, p_2, fpc_2 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 2, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_2 = np.argmax(u_2, axis=0)
cluster_membership_2_ =
pd.DataFrame(cluster_membership_2, columns=['Cluster 2'])
cluster_fuzz = cluster_membership_2_
chi_2 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_2
_)

cntr_3, u_3, u0_3, d_3, jm_3, p_3, fpc_3 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 3, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_3 = np.argmax(u_3, axis=0)
cluster_membership_3_ =
pd.DataFrame(cluster_membership_3, columns=['Cluster 3'])
cluster_fuzz['Cluster 3'] = cluster_membership_3_
chi_3 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_3
_)

cntr_4, u_4, u0_4, d_4, jm_4, p_4, fpc_4 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 4, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_4 = np.argmax(u_4, axis=0)
cluster_membership_4_ =
pd.DataFrame(cluster_membership_4, columns=['Cluster 4'])
cluster_fuzz['Cluster 4'] = cluster_membership_4_
chi_4 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_4
_)
```

```
cntr_5, u_5, u0_5, d_5, jm_5, p_5, fpc_5 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 5, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_5 = np.argmax(u_5, axis=0)
cluster_membership_5_ =
pd.DataFrame(cluster_membership_5, columns=['Cluster 5'])
cluster_fuzz['Cluster 5'] = cluster_membership_5_
chi_5 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_5
_)

cntr_6, u_6, u0_6, d_6, jm_6, p_6, fpc_6 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 6, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_6 = np.argmax(u_6, axis=0)
cluster_membership_6_ =
pd.DataFrame(cluster_membership_6, columns=['Cluster 6'])
cluster_fuzz['Cluster 6'] = cluster_membership_6_
chi_6 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_6
_)

cntr_7, u_7, u0_7, d_7, jm_7, p_7, fpc_7 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 7, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_7 = np.argmax(u_7, axis=0)
cluster_membership_7_ =
pd.DataFrame(cluster_membership_7, columns=['Cluster 7'])
cluster_fuzz['Cluster 7'] = cluster_membership_7_
chi_7 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_7
_)
```

```
cntr_8, u_8, u0_8, d_8, jm_8, p_8, fpc_8 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 8, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_8 = np.argmax(u_8, axis=0)
cluster_membership_8_ =
pd.DataFrame(cluster_membership_8, columns=['Cluster 8'])
cluster_fuzz['Cluster 8'] = cluster_membership_8_
chi_8 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_8
_)

cntr_9, u_9, u0_9, d_9, jm_9, p_9, fpc_9 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 9, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_9 = np.argmax(u_9, axis=0)
cluster_membership_9_ =
pd.DataFrame(cluster_membership_9, columns=['Cluster 9'])
cluster_fuzz['Cluster 9'] = cluster_membership_9_
chi_9 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_9
_)

cntr_10, u_10, u0_10, d_10, jm_10, p_10, fpc_10 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 10, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_10 = np.argmax(u_10, axis=0)
cluster_membership_10_ =
pd.DataFrame(cluster_membership_10, columns=['Cluster
10'])
cluster_fuzz['Cluster 10'] = cluster_membership_10_
chi_10 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
0_)
```

```
cntr_11, u_11, u0_11, d_11, jm_11, p_11, fpc_11 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 11, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_11 = np.argmax(u_11, axis=0)
cluster_membership_11_ =
pd.DataFrame(cluster_membership_11, columns=['Cluster
11'])
cluster_fuzz['Cluster 11'] = cluster_membership_11_
chi_11 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
1_)

cntr_12, u_12, u0_12, d_12, jm_12, p_12, fpc_12 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 12, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_12 = np.argmax(u_12, axis=0)
cluster_membership_12_ =
pd.DataFrame(cluster_membership_12, columns=['Cluster
12'])
cluster_fuzz['Cluster 12'] = cluster_membership_12_
chi_12 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
2_)

cntr_13, u_13, u0_13, d_13, jm_13, p_13, fpc_13 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 13, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_13 = np.argmax(u_13, axis=0)
cluster_membership_13_ =
pd.DataFrame(cluster_membership_13, columns=['Cluster
13'])
cluster_fuzz['Cluster 13'] = cluster_membership_13_
chi_13 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
3_)
```

```
cntr_14, u_14, u0_14, d_14, jm_14, p_14, fpc_14 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 14, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_14 = np.argmax(u_14, axis=0)
cluster_membership_14_ =
pd.DataFrame(cluster_membership_14, columns=['Cluster
14'])
cluster_fuzz['Cluster 14'] = cluster_membership_14_
chi_14 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
4_)

cntr_15, u_15, u0_15, d_15, jm_15, p_15, fpc_15 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 15, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_15 = np.argmax(u_15, axis=0)
cluster_membership_15_ =
pd.DataFrame(cluster_membership_15, columns=['Cluster
15'])
cluster_fuzz['Cluster 15'] = cluster_membership_15_
chi_15 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
5_)

cntr_16, u_16, u0_16, d_16, jm_16, p_16, fpc_16 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 16, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_16 = np.argmax(u_16, axis=0)
cluster_membership_16_ =
pd.DataFrame(cluster_membership_16, columns=['Cluster
16'])
cluster_fuzz['Cluster 16'] = cluster_membership_16_
```

```
chi_16 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
6_)

cntr_17, u_17, u0_17, d_17, jm_17, p_17, fpc_17 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 17, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_17 = np.argmax(u_17, axis=0)
cluster_membership_17_ =
pd.DataFrame(cluster_membership_17, columns=['Cluster
17'])
cluster_fuzz['Cluster 17'] = cluster_membership_17_
chi_17 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
7_)

cntr_18, u_18, u0_18, d_18, jm_18, p_18, fpc_18 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 18, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_18 = np.argmax(u_18, axis=0)
cluster_membership_18_ =
pd.DataFrame(cluster_membership_18, columns=['Cluster
18'])
cluster_fuzz['Cluster 18'] = cluster_membership_18_
chi_18 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
8_)

cntr_19, u_19, u0_19, d_19, jm_19, p_19, fpc_19 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 19, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_19 = np.argmax(u_19, axis=0)
cluster_membership_19_ =
pd.DataFrame(cluster_membership_19, columns=['Cluster
19'])
```

```
cluster_fuzz['Cluster 19'] = cluster_membership_19_
chi_19 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_1
9_)

cntr_20, u_20, u0_20, d_20, jm_20, p_20, fpc_20 =
skfuzzy.cluster.cmeans(TFIDF_fuzz, 20, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_20 = np.argmax(u_20, axis=0)
cluster_membership_20_ =
pd.DataFrame(cluster_membership_20, columns=['Cluster
20'])
cluster_fuzz['Cluster 20'] = cluster_membership_20_
chi_20 =
metrics.calinski_harabaz_score(TFIDF_,cluster_membership_2
0_)

cluster_fuzz.to_csv(r'Cluster Fuzzy UNIGRAM.csv')
print("For k=2, The CHI is {}".format(chi_2))
print("For k=3, The CHI is {}".format(chi_3))
print("For k=4, The CHI is {}".format(chi_4))
print("For k=5, The CHI is {}".format(chi_5))
print("For k=6, The CHI is {}".format(chi_6))
print("For k=7, The CHI is {}".format(chi_7))
print("For k=8, The CHI is {}".format(chi_8))
print("For k=9, The CHI is {}".format(chi_9))
print("For k=10, The CHI is {}".format(chi_10))
print("For k=11, The CHI is {}".format(chi_11))
print("For k=12, The CHI is {}".format(chi_12))
print("For k=13, The CHI is {}".format(chi_13))
print("For k=14, The CHI is {}".format(chi_14))
print("For k=15, The CHI is {}".format(chi_15))
print("For k=16, The CHI is {}".format(chi_16))
print("For k=17, The CHI is {}".format(chi_17))
print("For k=18, The CHI is {}".format(chi_18))
```

```
print("For k=19, The CHI is {}".format(chi_19))
print("For k=20, The CHI is {}".format(chi_20))

#Bigram
cluster_fuzz = pd.DataFrame()
cntr_2, u_2, u0_2, d_2, jm_2, p_2, fpc_2 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 2, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_2 = np.argmax(u_2, axis=0)
cluster_membership_2_ =
pd.DataFrame(cluster_membership_2, columns=['Cluster 2'])
cluster_fuzz = cluster_membership_2_
chi_2 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_2_)

cntr_3, u_3, u0_3, d_3, jm_3, p_3, fpc_3 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 3, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_3 = np.argmax(u_3, axis=0)
cluster_membership_3_ =
pd.DataFrame(cluster_membership_3, columns=['Cluster 3'])
cluster_fuzz['Cluster 3'] = cluster_membership_3_
chi_3 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_3_)

cntr_4, u_4, u0_4, d_4, jm_4, p_4, fpc_4 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 4, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_4 = np.argmax(u_4, axis=0)
cluster_membership_4_ =
pd.DataFrame(cluster_membership_4, columns=['Cluster 4'])
cluster_fuzz['Cluster 4'] = cluster_membership_4_
```

```
chi_4 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_4_)

cntr_5, u_5, u0_5, d_5, jm_5, p_5, fpc_5 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 5, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_5 = np.argmax(u_5, axis=0)
cluster_membership_5_ =
pd.DataFrame(cluster_membership_5, columns=['Cluster 5'])
cluster_fuzz['Cluster 5'] = cluster_membership_5_
chi_5 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_5_)

cntr_6, u_6, u0_6, d_6, jm_6, p_6, fpc_6 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 6, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_6 = np.argmax(u_6, axis=0)
cluster_membership_6_ =
pd.DataFrame(cluster_membership_6, columns=['Cluster 6'])
cluster_fuzz['Cluster 6'] = cluster_membership_6_
chi_6 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_6_)

cntr_7, u_7, u0_7, d_7, jm_7, p_7, fpc_7 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 7, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_7 = np.argmax(u_7, axis=0)
cluster_membership_7_ =
pd.DataFrame(cluster_membership_7, columns=['Cluster 7'])
cluster_fuzz['Cluster 7'] = cluster_membership_7_
```

```
chi_7 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_7_)

cntr_8, u_8, u0_8, d_8, jm_8, p_8, fpc_8 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 8, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_8 = np.argmax(u_8, axis=0)
cluster_membership_8_ =
pd.DataFrame(cluster_membership_8, columns=['Cluster 8'])
cluster_fuzz['Cluster 8'] = cluster_membership_8_
chi_8 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_8_)

cntr_9, u_9, u0_9, d_9, jm_9, p_9, fpc_9 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 9, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_9 = np.argmax(u_9, axis=0)
cluster_membership_9_ =
pd.DataFrame(cluster_membership_9, columns=['Cluster 9'])
cluster_fuzz['Cluster 9'] = cluster_membership_9_
chi_9 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_9_)

cntr_10, u_10, u0_10, d_10, jm_10, p_10, fpc_10 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 10, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_10 = np.argmax(u_10, axis=0)
cluster_membership_10_ =
pd.DataFrame(cluster_membership_10, columns=['Cluster
10'])
cluster_fuzz['Cluster 10'] = cluster_membership_10_
```

```
chi_10 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_10_)

cntr_11, u_11, u0_11, d_11, jm_11, p_11, fpc_11 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 11, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_11 = np.argmax(u_11, axis=0)
cluster_membership_11_ =
pd.DataFrame(cluster_membership_11, columns=['Cluster
11'])
cluster_fuzz['Cluster 11'] = cluster_membership_11_
chi_11 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_11_)

cntr_12, u_12, u0_12, d_12, jm_12, p_12, fpc_12 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 12, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_12 = np.argmax(u_12, axis=0)
cluster_membership_12_ =
pd.DataFrame(cluster_membership_12, columns=['Cluster
12'])
cluster_fuzz['Cluster 12'] = cluster_membership_12_
chi_12 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_12_)

cntr_13, u_13, u0_13, d_13, jm_13, p_13, fpc_13 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 13, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_13 = np.argmax(u_13, axis=0)
cluster_membership_13_ =
pd.DataFrame(cluster_membership_13, columns=['Cluster
13'])
```

```
cluster_fuzz['Cluster 13'] = cluster_membership_13_
chi_13 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_13_)

cntr_14, u_14, u0_14, d_14, jm_14, p_14, fpc_14 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 14, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_14 = np.argmax(u_14, axis=0)
cluster_membership_14_ =
pd.DataFrame(cluster_membership_14, columns=['Cluster
14'])
cluster_fuzz['Cluster 14'] = cluster_membership_14_
chi_14 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_14_)

cntr_15, u_15, u0_15, d_15, jm_15, p_15, fpc_15 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 15, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_15 = np.argmax(u_15, axis=0)
cluster_membership_15_ =
pd.DataFrame(cluster_membership_15, columns=['Cluster
15'])
cluster_fuzz['Cluster 15'] = cluster_membership_15_
chi_15 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_15_)

cntr_16, u_16, u0_16, d_16, jm_16, p_16, fpc_16 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 16, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_16 = np.argmax(u_16, axis=0)
```

```
cluster_membership_16_ =
pd.DataFrame(cluster_membership_16, columns=['Cluster
16'])
cluster_fuzz['Cluster 16'] = cluster_membership_16_
chi_16 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_16_)

cntr_17, u_17, u0_17, d_17, jm_17, p_17, fpc_17 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 17, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_17 = np.argmax(u_17, axis=0)
cluster_membership_17_ =
pd.DataFrame(cluster_membership_17, columns=['Cluster
17'])
cluster_fuzz['Cluster 17'] = cluster_membership_17_
chi_17 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_17_)

cntr_18, u_18, u0_18, d_18, jm_18, p_18, fpc_18 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 18, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_18 = np.argmax(u_18, axis=0)
cluster_membership_18_ =
pd.DataFrame(cluster_membership_18, columns=['Cluster
18'])
cluster_fuzz['Cluster 18'] = cluster_membership_18_
chi_18 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_18_)

cntr_19, u_19, u0_19, d_19, jm_19, p_19, fpc_19 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 19, 2, error=0.005,
maxiter=1000, init=None)
```

```
cluster_membership_19 = np.argmax(u_19, axis=0)
cluster_membership_19_ =
pd.DataFrame(cluster_membership_19, columns=['Cluster
19'])
cluster_fuzz['Cluster 19'] = cluster_membership_19_
chi_19 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_19_)

cntr_20, u_20, u0_20, d_20, jm_20, p_20, fpc_20 =
skfuzzy.cluster.cmeans(TFIDFBG_fuzz, 20, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_20 = np.argmax(u_20, axis=0)
cluster_membership_20_ =
pd.DataFrame(cluster_membership_20, columns=['Cluster
20'])
cluster_fuzz['Cluster 20'] = cluster_membership_20_
chi_20 =
metrics.calinski_harabaz_score(TFIDF_BG_,cluster_members
hip_20_)


cluster_fuzz.to_csv(r'Cluster Fuzzy BIGRAM.csv')
print("For k=2, The CHI is {}".format(chi_2))
print("For k=3, The CHI is {}".format(chi_3))
print("For k=4, The CHI is {}".format(chi_4))
print("For k=5, The CHI is {}".format(chi_5))
print("For k=6, The CHI is {}".format(chi_6))
print("For k=7, The CHI is {}".format(chi_7))
print("For k=8, The CHI is {}".format(chi_8))
print("For k=9, The CHI is {}".format(chi_9))
print("For k=10, The CHI is {}".format(chi_10))
print("For k=11, The CHI is {}".format(chi_11))
print("For k=12, The CHI is {}".format(chi_12))
print("For k=13, The CHI is {}".format(chi_13))
```

```
print("For k=14, The CHI is {}".format(chi_14))
print("For k=15, The CHI is {}".format(chi_15))
print("For k=16, The CHI is {}".format(chi_16))
print("For k=17, The CHI is {}".format(chi_17))
print("For k=18, The CHI is {}".format(chi_18))
print("For k=19, The CHI is {}".format(chi_19))
print("For k=20, The CHI is {}".format(chi_20))


#Trigram
cluster_fuzz = pd.DataFrame()
cntr_2, u_2, u0_2, d_2, jm_2, p_2, fpc_2 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 2, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_2 = np.argmax(u_2, axis=0)
cluster_membership_2_ =
pd.DataFrame(cluster_membership_2, columns=['Cluster 2'])
cluster_fuzz = cluster_membership_2_
chi_2 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_2_)

cntr_3, u_3, u0_3, d_3, jm_3, p_3, fpc_3 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 3, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_3 = np.argmax(u_3, axis=0)
cluster_membership_3_ =
pd.DataFrame(cluster_membership_3, columns=['Cluster 3'])
cluster_fuzz['Cluster 3'] = cluster_membership_3_
chi_3 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_3_)
```

```
cntr_4, u_4, u0_4, d_4, jm_4, p_4, fpc_4 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 4, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_4 = np.argmax(u_4, axis=0)
cluster_membership_4_ =
pd.DataFrame(cluster_membership_4, columns=['Cluster 4'])
cluster_fuzz['Cluster 4'] = cluster_membership_4_
chi_4 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_4_)

cntr_5, u_5, u0_5, d_5, jm_5, p_5, fpc_5 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 5, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_5 = np.argmax(u_5, axis=0)
cluster_membership_5_ =
pd.DataFrame(cluster_membership_5, columns=['Cluster 5'])
cluster_fuzz['Cluster 5'] = cluster_membership_5_
chi_5 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_5_)

cntr_6, u_6, u0_6, d_6, jm_6, p_6, fpc_6 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 6, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_6 = np.argmax(u_6, axis=0)
cluster_membership_6_ =
pd.DataFrame(cluster_membership_6, columns=['Cluster 6'])
cluster_fuzz['Cluster 6'] = cluster_membership_6_
chi_6 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_6_)
```

```
cntr_7, u_7, u0_7, d_7, jm_7, p_7, fpc_7 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 7, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_7 = np.argmax(u_7, axis=0)
cluster_membership_7_ =
pd.DataFrame(cluster_membership_7, columns=['Cluster 7'])
cluster_fuzz['Cluster 7'] = cluster_membership_7_
chi_7 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_7_)

cntr_8, u_8, u0_8, d_8, jm_8, p_8, fpc_8 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 8, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_8 = np.argmax(u_8, axis=0)
cluster_membership_8_ =
pd.DataFrame(cluster_membership_8, columns=['Cluster 8'])
cluster_fuzz['Cluster 8'] = cluster_membership_8_
chi_8 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_8_)

cntr_9, u_9, u0_9, d_9, jm_9, p_9, fpc_9 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 9, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_9 = np.argmax(u_9, axis=0)
cluster_membership_9_ =
pd.DataFrame(cluster_membership_9, columns=['Cluster 9'])
cluster_fuzz['Cluster 9'] = cluster_membership_9_
chi_9 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_9_)
```

```
cntr_10, u_10, u0_10, d_10, jm_10, p_10, fpc_10 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 10, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_10 = np.argmax(u_10, axis=0)
cluster_membership_10_ =
pd.DataFrame(cluster_membership_10, columns=['Cluster
10'])
cluster_fuzz['Cluster 10'] = cluster_membership_10_
chi_10 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_10_)

cntr_11, u_11, u0_11, d_11, jm_11, p_11, fpc_11 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 11, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_11 = np.argmax(u_11, axis=0)
cluster_membership_11_ =
pd.DataFrame(cluster_membership_11, columns=['Cluster
11'])
cluster_fuzz['Cluster 11'] = cluster_membership_11_
chi_11 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_11_)

cntr_12, u_12, u0_12, d_12, jm_12, p_12, fpc_12 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 12, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_12 = np.argmax(u_12, axis=0)
cluster_membership_12_ =
pd.DataFrame(cluster_membership_12, columns=['Cluster
12'])
cluster_fuzz['Cluster 12'] = cluster_membership_12_
chi_12 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_12_)
```

```
cntr_13, u_13, u0_13, d_13, jm_13, p_13, fpc_13 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 13, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_13 = np.argmax(u_13, axis=0)
cluster_membership_13_ =
pd.DataFrame(cluster_membership_13, columns=['Cluster
13'])
cluster_fuzz['Cluster 13'] = cluster_membership_13_
chi_13 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_13_)

cntr_14, u_14, u0_14, d_14, jm_14, p_14, fpc_14 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 14, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_14 = np.argmax(u_14, axis=0)
cluster_membership_14_ =
pd.DataFrame(cluster_membership_14, columns=['Cluster
14'])
cluster_fuzz['Cluster 14'] = cluster_membership_14_
chi_14 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_14_)

cntr_15, u_15, u0_15, d_15, jm_15, p_15, fpc_15 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 15, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_15 = np.argmax(u_15, axis=0)
cluster_membership_15_ =
pd.DataFrame(cluster_membership_15, columns=['Cluster
15'])
cluster_fuzz['Cluster 15'] = cluster_membership_15_
```

```
chi_15 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_15_)

cntr_16, u_16, u0_16, d_16, jm_16, p_16, fpc_16 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 16, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_16 = np.argmax(u_16, axis=0)
cluster_membership_16_ =
pd.DataFrame(cluster_membership_16, columns=['Cluster
16'])
cluster_fuzz['Cluster 16'] = cluster_membership_16_
chi_16 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_16_)

cntr_17, u_17, u0_17, d_17, jm_17, p_17, fpc_17 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 17, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_17 = np.argmax(u_17, axis=0)
cluster_membership_17_ =
pd.DataFrame(cluster_membership_17, columns=['Cluster
17'])
cluster_fuzz['Cluster 17'] = cluster_membership_17_
chi_17 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_17_)

cntr_18, u_18, u0_18, d_18, jm_18, p_18, fpc_18 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 18, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_18 = np.argmax(u_18, axis=0)
cluster_membership_18_ =
pd.DataFrame(cluster_membership_18, columns=['Cluster
18'])
```

```
cluster_fuzz['Cluster 18'] = cluster_membership_18_
chi_18 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_18_)

cntr_19, u_19, u0_19, d_19, jm_19, p_19, fpc_19 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 19, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_19 = np.argmax(u_19, axis=0)
cluster_membership_19_ =
pd.DataFrame(cluster_membership_19, columns=['Cluster
19'])
cluster_fuzz['Cluster 19'] = cluster_membership_19_
chi_19 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_19_)

cntr_20, u_20, u0_20, d_20, jm_20, p_20, fpc_20 =
skfuzzy.cluster.cmeans(TFIDFTG_fuzz, 20, 2, error=0.005,
maxiter=1000, init=None)
cluster_membership_20 = np.argmax(u_20, axis=0)
cluster_membership_20_ =
pd.DataFrame(cluster_membership_20, columns=['Cluster
20'])
cluster_fuzz['Cluster 20'] = cluster_membership_20_
chi_20 =
metrics.calinski_harabaz_score(TFIDF_TG_,cluster_members
hip_20_)



cluster_fuzz.to_csv(r'Cluster Fuzzy TRIGRAM.csv')
print("For k=2, The CHI is {}".format(chi_2))
print("For k=3, The CHI is {}".format(chi_3))
print("For k=4, The CHI is {}".format(chi_4))
print("For k=5, The CHI is {}".format(chi_5))
```

```
print("For k=6, The CHI is {}".format(chi_6))
print("For k=7, The CHI is {}".format(chi_7))
print("For k=8, The CHI is {}".format(chi_8))
print("For k=9, The CHI is {}".format(chi_9))
print("For k=10, The CHI is {}".format(chi_10))
print("For k=11, The CHI is {}".format(chi_11))
print("For k=12, The CHI is {}".format(chi_12))
print("For k=13, The CHI is {}".format(chi_13))
print("For k=14, The CHI is {}".format(chi_14))
print("For k=15, The CHI is {}".format(chi_15))
print("For k=16, The CHI is {}".format(chi_16))
print("For k=17, The CHI is {}".format(chi_17))
print("For k=18, The CHI is {}".format(chi_18))
print("For k=19, The CHI is {}".format(chi_19))
print("For k=20, The CHI is {}".format(chi_20))

# Fuzzy C-means For Optimal Number (K=3 TRIGRAM)
init_fpm = pd.DataFrame(u0_3).to_csv(r'E:\00.
Kuliah\SEMESTER 8\TUGAS AKHIR\LAPORAN\OLAHAN
FIX\FPM Initial.csv')
final_fpm = pd.DataFrame(u_3).to_csv(r'E:\00.
Kuliah\SEMESTER 8\TUGAS AKHIR\LAPORAN\OLAHAN
FIX\FPM Final.csv')
fungsi_obj = pd.DataFrame(jm_3).to_csv(r'E:\00.
Kuliah\SEMESTER 8\TUGAS AKHIR\LAPORAN\OLAHAN
FIX\FCM Obj Func.csv')
center = pd.DataFrame(cntr_3).to_csv(r'E:\00.
Kuliah\SEMESTER 8\TUGAS AKHIR\LAPORAN\OLAHAN
FIX\FCM Center.csv')

# number of iteration run
p_3
```

Lampiran 5. *Syntax* Frekuensi Tiap *Cluster* Metode Terbaik

```
# Frekuensi Cluster 1
resultTG_df_filter1 = resultTG_df.query('klaster<1')
resultTG_df_filter1.to_csv(r'E:\00. Kuliah\SEMESTER
8\TUGAS AKHIR\LAPORAN\filter1.csv')

df_filter1 = resultTG_df_filter1['text'].tolist()
df_filter1 = [word for line in df_filter1 for word in line.split()]

def ngrams(words, n):
    ngram_list = []
    for num in range(0, len(words)):
        ngram = '_'.join(words[num:num+n])
        ngram_list.append(ngram)
    return ngram_list
df_fil1 = ngrams(df_filter1, 3)

counts_fil1 = dict(Counter(df_fil1).most_common(10))
labels_fil1, values_fil1 = zip(*counts_fil1.items())

# sort values in descending order
indSort_fil1 = np.argsort(values_fil1)[::-1]

labels_fil1 = np.array(labels_fil1)[indSort_fil1]
values_fil1 = np.array(values_fil1)[indSort_fil1]
df_most1 = pd.DataFrame(labels_fil1, values_fil1)
df_most1

# Frekuensi Cluster 2
resultTG_df_filter2 = resultTG_df.query('0<klaster<2')
resultTG_df_filter2.to_csv(r'E:\00. Kuliah\SEMESTER
8\TUGAS AKHIR\LAPORAN\filter2.csv')

df_filter2 = resultTG_df_filter2['text'].tolist()
df_filter2 = [word for line in df_filter2 for word in line.split()]
```

```
def ngrams(words, n):
    ngram_list = []
    for num in range(0, len(words)):
        ngram = '_'.join(words[num:num+n])
        ngram_list.append(ngram)
    return ngram_list
df_fil2 = ngrams(df_filter2, 3)

counts_fil2 = dict(Counter(df_fil2).most_common(10))
labels_fil2, values_fil2 = zip(*counts_fil2.items())

# sort values in descending order
indSort_fil2 = np.argsort(values_fil2)[::-1]

labels_fil2 = np.array(labels_fil2)[indSort_fil2]
values_fil2 = np.array(values_fil2)[indSort_fil2]
df_most2 = pd.DataFrame(labels_fil2, values_fil2)
df_most2

# Frekuensi Cluster 3
resultTG_df_filter3 = resultTG_df.query('klaster>1')
resultTG_df_filter3.to_csv(r'E:\00. Kuliah\SEMESTER
8\TUGAS AKHIR\LAPORAN\filter3.csv')

df_filter3 = resultTG_df_filter3['text'].tolist()
df_filter3 = [word for line in df_filter3 for word in line.split()]

def ngrams(words, n):
    ngram_list = []
    for num in range(0, len(words)):
        ngram = '_'.join(words[num:num+n])
        ngram_list.append(ngram)
    return ngram_list
df_fil3 = ngrams(df_filter3, 3)
```

```
counts_fil3 = dict(Counter(df_fil3).most_common(10))
labels_fil3, values_fil3 = zip(*counts_fil3.items())

# sort values in descending order
indSort_fil3 = np.argsort(values_fil3)[::-1]

labels_fil3 = np.array(labels_fil3)[indSort_fil3]
values_fil3 = np.array(values_fil3)[indSort_fil3]
df_most3 = pd.DataFrame(labels_fil3, values_fil3)
df_most3
```

Lampiran 6. *Syntax Word Cloud* Menggunakan RStudio

```
library(word cloud)

data_DTM <- read.csv("MINDF0_TRIGRAM_DTM.csv",
sep=",")
klaster <- read.csv("CLUSTER KMEANS TRIGRAM 3.csv")

#Word cloud k=3
dtm3 <- data.frame(data_DTM, klaster$klaster)
dtm3 <- dtm3[-1]
for (i in 0:2){
  a <- dtm3[which(dtm3$klaster.klaster==i),-14790]
  b <- sort(colSums(a),decreasing=TRUE)
  c <- data.frame(word = names(b),freq=b)
  win.graph()
  word cloud(c$word, c$freq, scale=c(4,0.5), random.order=F,
min.freq=1, max.words=100, colors=brewer.pal(8, "Dark2"))
}
```

Lampiran 7. Surat Pernyataan Data

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FMKSD ITS,

Nama : Farizah Rizka Rahmaniar
NRP : 062115 4000 0111

menyatakan bahwa data yang digunakan dalam Tugas Akhir ini merupakan data sekunder yang diambil dari ~~penelitian / buku / Tugas Akhir / Thesis / Publikasi /~~ lainnya yaitu :

Sumber : Twitter API (*Application Program Interface*)
Keterangan : Data *tweet* dengan *keyword* "@KAI121" dari tanggal 8 – 18 Maret 2019

Surat pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data maka saya siap menerima sanksi sesuai aturan yang berlaku.

Surabaya, Mei 2019

Mengetahui,
Pembimbing Tugas Akhir

Dr. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002

Mahasiswa

Farizah Rizka Rahmaniar
NRP. 062115 4000 0111

BIODATA PENULIS

Penulis dilahirkan di Sidoarjo, 14 Juni 1997 dengan nama lengkap Farizah Rizka Rahmaniar, biasa dipanggil Risa. Penulis menempuh pendidikan formal di SD Muhammadiyah 5 Porong (2003-2009), SMPN 1 Sidoarjo (2009-2012), SMAN 1 Sidoarjo (2012-2015), dan melanjutkan studinya di Program Studi Sarjana Departemen Statistika, Fakultas Matematika, Komputasi, dan Sains Data (FMKSD) ITS. Selama masa perkuliahan, penulis aktif dalam organisasi Himpunan Mahasiswa Statistika ITS (HIMASTA-ITS) sebagai staff Departemen Sosial periode 2016-2017 dan Sekretaris Departemen Sosial dan Masyarakat periode 2017-2018. Selain itu, penulis juga aktif dalam kepanitiaan, seperti Pekan Raya Statistika (PRS) tahun 2017 dan 2018, Generasi Integralistik (GERIGI) tahun 2016 dan 2017, dan Bina Cinta Statistika (BCS) tahun 2016 dan 2017. Bagi pembaca yang ingin berdiskusi, memberikan saran, dan kritik mengenai Tugas Akhir ini dapat disampaikan melalui email farizahrizka@gmail.com.