KERJA PRAKTIK – IF184801

Pembuatan Kamus *Dangerous Speech* Berdasarkan Aspek

Teknik Informatika Institut Teknologi Sepuluh Nopember
Jalan Teknik Kimia, Keputih, Kec. Sukolilo, Kota Surabaya

Periode: 3 Februari 2023 – 3 April 2023

Oleh:

Christian Bennett Robin 05111940000078

Pembimbing Jurusan
Siska Arifiani, S.Kom., M.Kom.

Pembimbing Lapangan
Yulian Findawati, S.T., M.MT.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2023

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – IF184801

Pembuatan Kamus *Dangerous Speech* Berdasarkan Aspek

Teknik Informatika Institut Teknologi Sepuluh Nopember
Jalan Teknik Kimia, Keputih, Kec. Sukolilo, Kota Surabaya

Periode: 3 Februari 2023 – 3 April 2023

Oleh:

Christian Bennett Robin 05111940000078

Pembimbing Jurusan
Siska Arifiani, S.Kom., M.Kom.

Pembimbing Lapangan
Yulian Findawati, S.T., M.MT.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2023

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

Pembuatan Kamus Dangerous Speech Berdasarkan Aspek

Oleh:

Christian Bennett Robin 05111940000078

Mengetahui,	Menyetujui,
Pembimbing Kerja Praktik	Dosen Pembimbing Kerja Praktik
	
Yulian Findawati, S.T., M.MT.	Siska Arifiani, S.Kom., M.Kom.
NIDN: 0725078301	NIP: 1990202012034

SURABAYA, April 2023

Pembuatan Kamus *Dangerous Speech* Berdasarkan Aspek

Nama Mahasiswa	**: Christian Bennett Robin**
NRP	**: 05111940000078**
Departemen	**: Teknik Informatika**
Pembimbing Jurusan	**: Siska Arifiani, S.Kom., M.Kom.**
Pembimbing Lapangan	**: Yulian Findawati, S.T., M.MT.**

ABSTRAK

Dangerous speech, atau ujaran berbahaya, yang merupakan bagian dari ujaran kebencian, merupakan suatu ujaran yang dapat meningkatkan risiko seseorang atau suatu kelompok orang melakukan kejahatan terhadap orang lain atau kelompok orang lainnya. Dalam *dangerous speech* ini, terdapat 2 aspek konteks, yaitu sosial dan historis, serta 5 aspek pesan, yaitu dehumanisasi, tuduhan, serangan terhadap wanita dan anak-anak, loyalitas kelompok, dan ancaman terhadap suatu kelompok. *Dangerous speech* sendiri marak ditemukan di media sosial seperti *Twitter*. Pada pengerjaan Kerja Praktik ini, penulis telah membuat suatu *kamus* yang berisikan kata-kata kunci untuk masing-masing aspek *dangerous speech*. *Kamus* tersebut berbentuk suatu *file* excel dan memuat kata-kata *n-gram* untuk setiap aspek *dangerous speech*, serta pola kalimat untuk *n-gram* yang paling sering muncul dengan cara *Part-of-speech (POS) tagging* yang dapat dianalisis lebih lanjut dan dipergunakan seperlunya pada penelitian kedepannya.

Kata kunci: Dangerous speech, n-gram, Part-of-speech (POS) tagging

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya penulis dapat melaksanakan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika, yakni Kerja Praktik (KP).

Penulis menyadari masih ada kekurangan baik dalam pelaksanaan Kerja Praktik maupun penyusunan buku laporan ini. Namun, penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan Kerja Praktik ini.

Melalui buku laporan ini, penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik secara langsung maupun tidak langsung dalam pelaksanaan Kerja Praktik hingga penyusunan buku laporan ini. Orang-orang tersebut antara lain adalah:

1. Kedua orang tua penulis.
2. Bapak Ary Mazharuddin Shiddiqi S.Kom., M.Comp., Ph.D, selaku koordinator Kerja Praktik.

3. Ibu Siska Arifiani, S.Kom., M.Kom., selaku pembimbing jurusan Teknik Informatika.
4. Ibu Yulian Findawati, S.T., M.MT., selaku pembimbing lapangan Kerja Praktik.

Surabaya, 5 April 2023

Christian Bennett Robin

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Twitter merupakan salah satu *platform* media sosial yang paling banyak digunakan di Indonesia, dengan jumlah pengguna yang mencapai lebih dari 18,45 juta orang pada tahun 2022, yang menempati peringkat ke-5 pengguna *Twitter* terbanyak di dunia (Statista, 2022). Dengan begitu banyak orang yang terhubung melalui *platform* ini, *Twitter* menjadi tempat yang ideal bagi para penggunanya untuk berbagi pendapat, berdiskusi, dan menyebarkan informasi. Namun, juga merupakan tempat yang rawan terjadinya penyebaran *dangerous speech* atau ujaran berbahaya.

Dangerous speech sendiri atau ujaran berbahaya adalah segala bentuk ekspresi (seperti ucapan, teks, ataupun gambar) yang dapat meningkatkan risiko bahwa audiensnya akan membenarkan atau melakukan kekerasan terhadap orang lain atau bahkan kelompok lain. Definisi tersebut lebih mengacu pada peningkatan risiko terjadinya kekerasan, bukan sebagai penyebab kekerasan itu sendiri. Secara umum, tema yang

sering muncul dalam konteks *dangerous speech* adalah bagaimana orang-orang dari kelompok lain dianggap sebagai ancaman yang begitu serius sehingga kekerasan terhadap mereka dapat diterima atau bahkan diperlukan.

Di Indonesia, masalah penyebaran *dangerous speech* telah menjadi isu serius yang dapat menyebabkan terjadinya konflik dan kekerasan sosial. Banyak kasus telah dilaporkan terkait penyebaran *dangerous speech* melalui media sosial, terutama di *Twitter*, seperti yang terjadi pada akun @ibhaskiss yang ditangkap pada Februari 2018 karena menyebarkan konten *dangerous speech* yang juga berhubungan dengan Suku, Agama, dan Ras (SARA). Walaupun begitu, masih saja terdapat konten *dangerous speech* yang beredar di media sosial hingga saat ini.

Pembuatan kamus yang berisikan kata-kata kunci berdasarkan 7 aspek *dangerous speech* menurut (Benesch et al., 2018) dapat memudahkan analisa lebih lanjut terhadap penelitian *dangerous speech* yang dilakukan oleh Ibu Yulian Findawati, S.T., M.MT.. Kamus tersebut dapat dianalisis lebih lanjut dan dilakukan pembuatan *dataset* berbasis *keyword* untuk dianalisa lebih lanjut menggunakan model klasifikasi *machine learning*.

1.2. Tujuan

Tujuan dari Kerja Praktik ini adalah untuk menyelesaikan kewajiban kuliah Kerja Praktik di Institut Teknologi Sepuluh Nopember dengan bobot 2 (dua) SKS. Selain itu, tujuan lain dari pembuatan kamus ini adalah agar hasilnya dapat dianalisis lebih lanjut dan digunakan sesuai dengan keperluan kedepannya.

1.3. Manfaat

Dengan kamus yang dibuat, diharapkan hasil dari kamus ini dapat digunakan untuk dianalisis lebih lanjut dan agar dapat bermanfaat bagi penelitian kedepannya.

1.4. Rumusan Masalah

Berikut ini rumusan masalah pada pembuatan kamus ini:

1. Bagaimana proses pengambilan serta pembersihan data *tweet* dari *Twitter*?
2. Bagaimana cara melakukan pelabelan data *tweet* yang bersih guna untuk memperoleh *dataset?*
3. Bagaimana cara mengekstrak *n-gram* serta frekuensi masing-masing *n-gram* tersebut dari *dataset* yang sudah diperoleh?

4. Bagaimana cara menentukan pola kalimat *n-gram* yang telah diperoleh?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi	: *Online*
Waktu	: 3 Februari – 3 April 2023
Hari Kerja	: Senin – Jumat
Jam Kerja	: Fleksibel

1.6. Metodologi Kerja Praktik

Tahapan pengerjaan kerja praktik ini dapat dijabarkan sebagai berikut:

1.6.1. Perumusan Masalah

Tahapan ini meliputi eksplorasi terhadap permasalahan yang ada. Segala kebutuhan terhadap permasalahan tersebut juga dieksplorasi.

1.6.2. Studi Literatur

Pada tahapan ini, dilakukan proses pencarian, pembelajaran, dan pengumpulan informasi yang berkaitan dengan implementasi sistem yang akan dibuat. Informasi

dapat diperoleh dari internet ataupun dari proyek sebelumnya yang serupa dan memungkinkan untuk diimplementasikan.

1.6.3. Analisis dan Perancangan

Tahapan ini meliputi analisis mengenai hasil dari studi literatur yang dilakukan. Dari beberapa metode yang ditemukan saat studi literatur, dianalisa metode mana yang paling tepat dan efektif untuk digunakan untuk menyelesaikan permasalahan.

1.6.4. Implementasi Sistem

Pada tahapan ini, dilakukan implementasi sistem hasil analisa sebelumnya. Sistem ini menggunakan Google Colab sebagai *Integrated Development Environment* (*IDE*)-nya dan Python sebagai bahasa-nya.

1.6.5. Pengujian dan Evaluasi

Tahapan ini meliputi pengujian terhadap *kamus* yang telah dibuat. Pengujian berupa analisa hasil kamus oleh Ibu Yulian Findawati, S.T., M.MT.

1.6.6. Kesimpulan dan Saran

Pada tahapan ini, dipaparkan kesimpulan yang dapat diambil dan juga saran selama pengerjaan kerja praktik ini.

1.7. Sistematika Laporan

Laporan kerja praktik ini terdiri dari tujuh bab dengan rincian sebagai berikut:

1.7.1. Bab I Pendahuluan

Pada bab ini, dijelaskan tentang latar belakang, tujuan, manfaat, rumusan permasalahan, waktu pelaksanaan, serta metodologi dan sistematika pengerjaan kerja praktik dan juga penulisan laporan kerja praktik.

1.7.2. Bab II Profil Perusahaan

Bab ini akan menjelaskan secara rinci tentang profil Departemen Teknik Informatika ITS.

1.7.3. Bab III Tinjauan Pustaka

Pada bab ini, dijelaskan mengenai tinjauan pustaka dan literatur yang digunakan dalam pelaksanaan kerja praktik.

1.7.4. Bab IV Analisis dan Perancangan Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaik

1.7.5. Bab V Implementasi Sistem

Bab ini berisi penjelasan tentang tahapan implementasi sistem yang dilakukan.

1.7.6. Bab VI Pengujian dan Evaluasi

Pada bab ini, dijelaskan tentang hasil pengujian dan evaluasi dari kamus yang telah dibuat.

1.7.7. Bab VII Kesimpulan dan Saran

Pada bab ini, akan dipaparkan kesimpulan yang dapat diambil dan juga saran selama pengerjaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil Departemen Teknik Informatika ITS

Sebagai perkiraan untuk saat ini dan masa yang akan datang, teknologi informasi menjadi tulang punggung pertumbuhan ekonomi bangsa. Saat ini pun invasi teknologi informasi sudah terasa di berbagai bidang kehidupan manusia. Hal ini sepenuhnya disadari oleh pemerintah, sehingga sejak Repelita V yang lalu, pemerintah telah mencanangkan bahwa pengembangan pendidikan tinggi dalam bidang komputer dan informatika merupakan salah satu program prioritas, bersama-sama dengan disiplin ilmu lainnya seperti rekayasa, perilaku, manajemen, akuntansi, dan kesenian.

Pendidikan tinggi diarahkan untuk mempersiapkan bangsa Indonesia dalam menghadapi era pembangunan industri dan informasi. Untuk itu pemerintah melalui Direktorat Jendral Pendidikan Tinggi pada tahun 1985 menginstruksikan untuk membuka Program Studi S1 baru untuk bidang ilmu teknologi komputer di empat universitas atau institut di mana ITS termasuk di dalamnya. Di ITS, program ini awalnya diberi nama Program Studi Teknik Komputer. Namun sejak tahun 1993, nama Program

Studi Teknik Komputer diubah menjadi Jurusan Teknik Komputer. Akhirnya, pada tahun 1996 secara resmi jurusan ini berganti nama menjadi Jurusan Teknik Informatika berdasarkan Surat Keputusan Direktur Jendral Pendidikan Tinggi Nomor 224/DIKTI/Kep/1996, tanggal 11 Juli 1996. Pada saat ini, Jurusan Teknik Informatika memperoleh nilai akreditasi A berdasarkan Surat Keputusan Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT) Nomor 003/BAN-PT/Ak-X/S1/V/2006, tanggal 18 Mei 2006.

Selain program Sarjana (S1), Jurusan Teknik Informatika juga menyelenggarakan program Pasca Sarjana (S2) yang dirintis sejak tahun 1994, dengan surat keputusan Direktur Jendral Pendidikan Tinggi No. 2851/D/T/2001, perihal ijin penyelenggaraan Program-Program Studi Jenjang Program Strata-2 (S2) pada Institut Teknologi Sepuluh Nopember Surabaya. Dan pada tahun 2011, Jurusan Teknik Informatika mulai menyelenggarakan program Doktor (S3).

2.2. Visi dan Misi Perusahaan

2.2.1. Visi

Sejalan dengan visi ITS yaitu menjadi perguruan tinggi dengan reputasi internasional dalam ilmu pengetahuan, teknologi, dan seni, terutama yang menunjang industri dan

kelautan yang berwawasan lingkungan, maka visi Departemen Informatika adalah menjadi inovator bidang informatika yang unggul di tingkat nasional dengan reputasi internasional, serta berperan aktif dalam upaya memajukan dan mensejahterakan bangsa.

Visi PSTI adalah menjadi lembaga pendidikan dan penelitian di bidang informatika yang unggul di tingkat nasional dan memiliki reputasi internasional.

2.2.2. Misi

Memberi layanan prima dan solusi untuk pekerjaan secara remote, agar dapat menjadi produktif bagi para pihak yang terlibat.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. *Dangerous speech* berdasarkan aspek

Menurut penelitian (Benesch et al., 2018), *dangerous speech* terdiri dari 7 aspek konteks dan pesan. Aspek konteks sendiri terdiri dari 2 konteks, yaitu konteks sosial dan historis, sedangkan untuk aspek pesan terdiri dari 5 konteks, yaitu dehumanisasi (*dehumanization)*, tuduhan (*accusation in a mirror)*, serangan terhadap wanita dan anak kecil (*assertion of attack against women and girls)*, mempertanyakan loyalitas suatu kelompok (*questioning in-group loyalty)*, dan ancaman terhadap suatu kelompok (*threat to group integrity or purity*).

Aspek sosial merupakan aspek yang berkaitan erat dengan kesejahteraan manusia yang terkait, seperti misalnya pendidikan, bencana yang sedang terjadi, situasi politik yang sedang berlangsung, dan lainnya, sedangkan aspek historis berkaitan erat dengan peristiwa-peristiwa sejarah yang pernah terjadi, seperti misalnya peristiwa tragedi Kanjuruhan. Dehumanisasi merupakan aspek dimana seseorang atau suatu kelompok orang dianggap lebih rendah dari manusia, dan oleh karena itu, orang-orang tidak akan segan untuk melakukan kejahatan terhadap

orang atau kelompok tersebut. Tuduhan adalah suatu aspek dimana suatu kejahatan atau tindakan buruk yang tidak dilakukan oleh suatu pihak malah dituduhkan ke pihak tersebut. Serangan terhadap wanita dan anak kecil merupakan aspek dimana terdapat perlakuan tidak mengenakan kepada wanita atau anak kecil. Perlakuan ini dapat berupa ancaman, gangguan, atau bahkan pelecehan. Mempertanyakan loyalitas suatu kelompok merupakan aspek dimana seseorang dalam suatu kelompok malah menargetkan anggota dalam kelompoknya dan menuduh kalau anggota tersebut kurang setia pada kelompoknya, atau bahkan dicap sebagai pengkhianat dalam kelompoknya. Ancaman terhadap suatu kelompok merupakan aspek dimana seseorang melakukan provokasi terhadap kelompoknya sendiri sehingga mereka berpikir bahwa ada kelompok lain yang akan melakukan serangan terhadap kelompok mereka. Hal ini dapat membuat anggota kelompok yang terprovokasi tersebut malah menjadi defensif dan memaklumkan perbuatan jahat yang bisa saja dilakukan mereka sendiri dengan alasan pembelaan diri (Benesch et al., 2018).

3.2. Penggunaan *SNScrape* untuk kebutuhan pengumpulan data dari *Twitter*

SNScrape adalah sebuah *Command Line Interface* (*CLI*) atau alat baris perintah yang memungkinkan pengambilan data dari *platform* media sosial seperti *Twitter* dan *Instagram*. Alat ini dirancang untuk mudah digunakan dan memberikan banyak data dalam waktu singkat. Dengan *SNScrape*, dapat ditentukan jenis data yang ingin dikumpulkan (misalnya, *tweet*, komentar, suka, dll.), serta *filter* atau *parameter* yang relevan (misalnya, *hashtag*, *ID* pengguna, dll.). Data kemudian dapat disimpan ke dalam suatu *file* berbentuk .csv untuk dianalisis lebih lanjut.

SNScrape ditulis dalam bahasa *Python* dan dapat diunduh menggunakan manajer paket *Python pip*. Ini adalah perangkat lunak sumber terbuka, yang berarti bahwa kode sumber tersedia untuk dilihat dan dimodifikasi oleh siapa saja.

Selain sebagai *CLI*, *SNScrape* juga dapat digunakan sebagai *library* walaupun belum ada dokumentasinya di *GitHub*-nya. Pada pengerjaan kerja praktik ini, *SNScrape* akan digunakan sebagai *library* untuk mempermudah proses pengumpulan data agar dapat diatur parameter pengambilan datanya (JustAnotherArchivist, 2018/2022).

3.3. Pra-proses data teks

Pra-proses data merupakan proses pembersihan dan persiapan data untuk proses klasifikasi. Data teks online biasanya mengandung banyak *noise* dan kata-kata yang tidak informatif seperti misalnya *tag HTML*. Selain itu, terdapat banyak kata dalam teks yang tidak memiliki arti atau tidak bermakna. Hal tersebutlah yang membuat data teks sebaiknya dilakukan pra-proses untuk mendapatkan data yang bersih.

Melakukan pra-proses teks dianggap sebagai suatu langkah yang penting untuk melakukan tugas *Natural Language Processing* (*NLP*). Penelitian dari (Kurniasih & Manik, 2022) menyatakan bahwa pra-proses data teks dapat secara signifikan mempengaruhi performa model tersebut, dan mereka mengevaluasi efektivitas beberapa teknik pra-proses teks yang berbeda. Penelitian tersebut menemukan bahwa teknik sederhana seperti melakukan *case-folding* dan menghilangkan tanda baca dapat meningkatkan performa model, sementara teknik lainnya seperti *stemming* dan *lemmatization* tidak berpengaruh begitu signifikan terhadap performa model.

3.4. N-*gram* dalam teks

N-gram adalah sekelompok kata dalam urutan teks atau ucapan. Sebagian besar *n-gram* diekstraksi dari korpus teks atau

ucapan. *N-gram* adalah urutan *N* kata, misalnya, *1-gram* (atau *unigram*) yang berasal dari kata “tolong” atau “ambil”, lalu *2-gram* (atau *bigram*) berasal dari kata-kata seperti "tolong ambil", "ambil pensil", dan *3-gram* (atau *trigram*) adalah urutan tiga kata seperti "tolong ambil pensil" (Zhu et al., 2022).

3.5. Menentukan pola kalimat dengan *Part-of-speech (POS) tagging*

Part-of-speech (*POS*) *tagging*, juga disebut sebagai penanda gramatikal, adalah pemberian otomatis tanda-tanda *POS* pada kata-kata dalam sebuah kalimat. *POS* adalah jenis klasifikasi gramatikal yang umumnya mencakup kata kerja, kata sifat, kata keterangan, kata benda, dan lain-lain. *POS tagging* adalah aplikasi pemrosesan bahasa alami yang penting yang digunakan dalam terjemahan mesin, mengartikan makna kata, analisa sintaksis, dan sebagainya. Awal mula *POS tagging* didasarkan pada ambiguitas banyak kata dalam hal bagian bicara mereka dalam sebuah konteks (Chiche & Yitagesu, 2022).

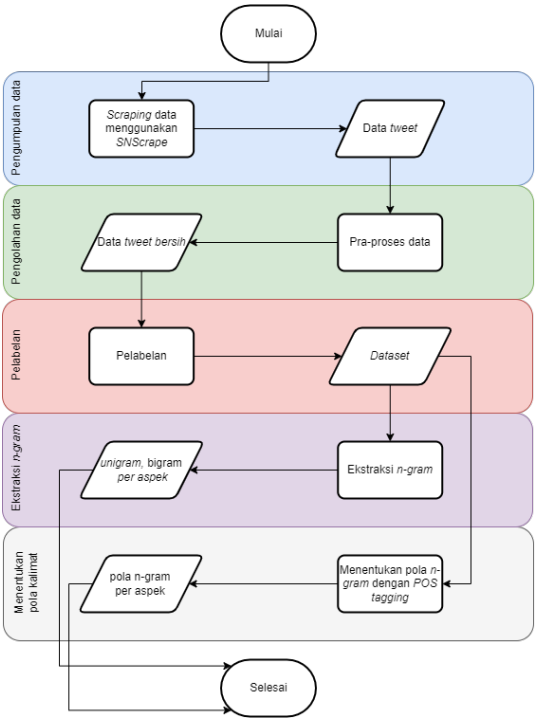
[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1. Rancangan sistem

Perancangan sistem ini mencakup pengumpulan data, pengolahan data, pelabelan, ekstraksi *n-gram*, dan yang terakhir, yaitu menentukan pola kalimat.

Berikut merupakan diagram alir untuk perancangan sistem:



Gambar 4.1.1 Diagram alir sistem

4.2. Pengumpulan data

Pengumpulan data dilakukan menggunakan *SNScrape* dan dilakukan untuk akun *Twitter* @ruhutsitompul. Berikut merupakan atribut-atribut *tweet* yang akan dikumpulkan dari *Twitter* serta penjelasannya*:*

Tabel 4.2.1 Penjelasan atribut tweet

Atribut	**Penjelasan**
ID	*ID* dari tweet
Tweet Time	Waktu *tweet* diunggah
Conversation ID	*ID* dari percakapan
Likes Count	Jumlah banyak *like*
Retweets Count	Jumlah banyak *retweet*
Tweets	Isi dari *tweet*
in reply to Tweet ID	*ID* dari tweet yang sedang dibalas
in reply to	*ID* dari pengguna yang *tweet*-nya sedang dibalas
User	*Username* pengguna
User Location	Lokasi pengguna
Followers Count	Jumlah pengikut pengguna
Followings Count	Jumlah mengikuti pengguna

Likes Count	Jumlah total *like* akun pengguna
Tweets Count	Jumlah total *tweet* akun pengguna

4.3. Pengolahan data

Data akan diolah dan pra-proses dengan berbagai metode sebagai berikut:

1. *Case folding*
2. Mengubah entitas *HTML* jika memang ada (*'&'*, *'<'*, *'>'*)
3. Mengubah *hashtags* dan *usernames* yang sering muncul menjadi bentuk panjang dan aslinya (contoh: '#debatfinalpilkadajkt' menjadi 'debat final pilkada jakarta', '@basukibtp' menjadi 'ahok', dan seterusnya)
4. Mengubah awalan *link* menjadi '*HTTPURL'*
5. Menghapus *HTML tag*
6. Menghapus tanda *hashtags* pada *hashtags* yang tidak sering muncul
7. Mengubah *mention usernames* yang tidak sering muncul menjadi '*@USER'*
8. Menghapus karakter yang bukan huruf
9. Menghapus spasi yang berlebihan
10. Melakukan *stopwords removal*

11. Melakukan *lemmatizing*
12. Membuang data *tweet* jika jumlah kata di *tweet* tersebut dibawah 4

4.4. Pelabelan

Data yang telah diolah selanjutnya dilakukan proses pelabelan secara manual guna untuk memperoleh *dataset* yang dapat digunakan untuk proses selanjutnya. Pelabelan manual dilakukan berdasarkan pemahaman penulis terkait masing-masing aspek *dangerous speech* berdasarkan diskusi dengan Ibu Yulian Findawati S.T., M.MT.

4.5. Ekstraksi *n-gram*

Setelah *dataset* sudah diperoleh setelah dilabeli, selanjutnya dilakukan ekstraksi *n-gram.* Ekstraksi *n-gram* dilakukan untuk *unigram* dan *bigram* untuk semua aspek *dangerous speech*, dan hasilnya dimasukkan kedalam sebuah *file excel* agar dapat dianalisa dengan lebih mudah.

4.6. Menentukan pola *n-gram* dengan *Part-of-speech (POS) tagging*

Setelah melakukan ekstraksi *n-gram*, selanjutnya ditentukan pola kalimat dari masing-masing *n-gram* dengan cara *POS tagging* dengan *library Flair*. Penentuan pola kalimat ini

dilakukan untuk setiap *n-gram* yang telah didapatkan dan hasilnya dimasukkan ke dalam sebuah *file excel* agar dapat dianalisa dengan lebih mudah.

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

5.1. Pengumpulan data

Hal pertama yang dilakukan adalah pengumpulan data *Twitter* akun @ruhutsitompul menggunakan *SNScrape*. Berikut merupakan potongan kode untuk proses pengumpulan data:

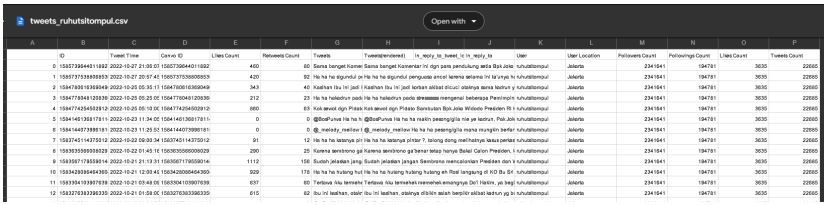
```
import snscrape.modules.twitter as sntwitter
import pandas as pd
from datetime import datetime
from datetime import timedelta
import pytz
import re

utc=pytz.UTC
tweet_container = []
attribut_container = []

for i, tweet in enumerate(sntwitter.TwitterSearchScraper('since:2019-04-17 until:2023-04-17 from:ruhutsitompul').get_items()):
    if tweet.conversationId == tweet.id and tweet.user.username == 'ruhutsitompul':
        get_convo(tweet.id)

tweets_ruhutsitompul = pd.DataFrame(tweet_container, columns=["ID", "Tweet Time", "Convo ID", "Likes Count", "Retweets Count",
                                                              "Tweets", 'Tweets(rendered)',"in_reply_to_tweet_id", "in_reply_to",
                                                              "User", "User Location", "Followers Count", "Followings Count",
                                                              "Likes Count", "Tweets Count"])
```

```
0
https://twitter.com/03__nakula/status/1589588116032978944
https://twitter.com/maheswararajen1/status/1589590768901324800
1
2
3
https://twitter.com/03__nakula/status/1589570952509079553
4
5
https://twitter.com/03__nakula/status/1589559516865011715
https://twitter.com/03__nakula/status/1589559519742263297
6
https://twitter.com/03__nakula/status/1589558597125435392
7
8
https://twitter.com/03__nakula/status/1589523060121825281
https://twitter.com/RekjanS/status/1589524882454646785
https://twitter.com/Rudatln7/status/1589525800696807430
https://twitter.com/doirx/status/1589528119836553218
```

Gambar 5.1.1 Potongan kode SNScrape

Hasil akhir dari proses pengumpulan data pada Gambar 5.1.1 adalah sebuah *file .csv* seperti pada Gambar 5.1.2 yang berisi *ID, Tweet Time, Convo ID, Likes Count, Followers Count, Tweets, Tweets(rendered), in_reply_to_tweet_id, in_reply_to, User, User*

Location, Followers Count, Followings Count, Likes Count, dan Tweets Count.



Gambar 5.1.2 Data .csv hasil pengumpulan data

5.2. Pengolahan data

Tahapan selanjutnya adalah pengolahan data. Data dipra-proses dengan berbagai metode yang dijabarkan pada Bab 4.3. Gambar 5.2.1 dan Gambar 5.2.2 merupakan potongan kode untuk proses pengolahan data yang mencakup *case-folding*, penggantian entitas *HTML* seperti *&amp*, *&gt*, dan *&lt*, penghapusan *URL,* tag HTML, simbol dan spasi berlebih, penggantian *mention* menjadi *@USER*, serta penghapusan *stopword*. Selain itu, kalimat yang jumlah katanya dibawah 4 akan dihapus.

```
[ ] import gensim
    import re
    from nlp_id.tokenizer import Tokenizer
    tokenizer = Tokenizer()

    # Function to preprocess a text
    def preprocess(text):
      # Case folding
      text = text.lower()

      # Change HTML entities
      text = text.replace('&amp;', 'dan')
      text = text.replace('&gt;', 'lebih dari')
      text = text.replace('&lt;', 'kurang dari')

      # Change top occuring hashtags and mentions
      for word, replacement in top_hashtags_usernames.items():
        text = text.replace(word, replacement)

      # Remove url
      text = re.sub(r'http\S+', 'HTTPURL', text)

      # Remove HTML tags
      text = re.sub(r'<.*?>', ' ', text)
```

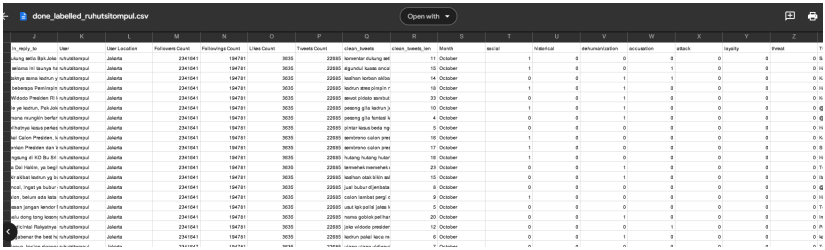
Gambar 5.2.1 Data pre-processing 1

```
    # Remove hashtags
    text = re.sub(r'#\w+', ' ', text)

    # Replace @mentions with '@USER'
    text = re.sub(r'@\w+', '@USER', text)

    # Remove non-letter characters
    text = re.sub('[^a-zA-z]', ' ', text)

    # Remove excess space
    text = re.sub(' +', ' ', text)
    text = text.strip()

    result = []
    word_token = tokenizer.tokenize(text) # Tokenize words
    for word in word_token:
      word = word.strip().lower() # Case Folding to Lower Case
      if word in data_slang:
        word = data_slang[word]
      if word not in data_stopword: # Stopwords removal
        result.append(word)
      else:
        continue

    if(len(result) < 4):
      result_sentence = None  # Drop words with less than 4 characters
    else:
      result_sentence = " ".join(result).strip() # Concatenate the result of preprocessing

    return result_sentence
```

Gambar 5.2.2 Data pre-processing 2

5.3. Pelabelan

Setelah data *tweet* sudah dipra-proses dan bersih, selanjutnya dilakukan proses pelabelan guna untuk memperoleh dataset awal. Pelabelan dilakukan secara manual berdasarkan pengertian dari 7 aspek *dangerous speech* dari Benesch et al oleh penulis. Hasil akhir dari proses pelabelan ini berupa sebuah *file .csv* pada Gambar 5.3.1.



Gambar 5.3.1 Data tweet hasil pelabelan

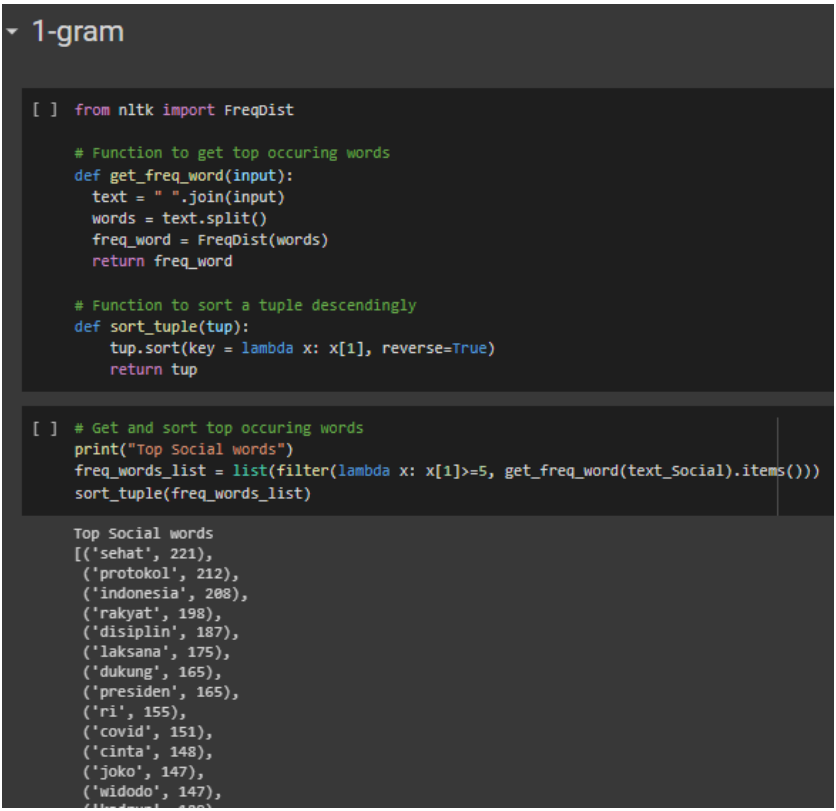
5.4. Ekstraksi *n-gram*

Setelah *dataset* sudah diperoleh dari hasil pelabelan, tahapan selanjutnya adalah melakukan ekstraksi *n-gram* dari *dataset* tersebut. *N-gram* yang diekstrak adalah *unigram* dan *bigram*. Gambar 5.4.1 hingga 5.4.15 merupakan beberapa potongan kode untuk ekstraksi *unigram* dan *bigram* dari *dataset* yang sudah dilabel.

Untuk *unigram,* yang dapat dilihat dari Gambar 5.4.1 hingga 5.4.7, pertama-tama didefinisikan fungsi *get_freq_word(input)* untuk mendapatkan frekuensi dari setiap kata dari sebuah masukan kolom *dataframe*. Fungsi ini menggunakan fungsi dari *library nltk* yaitu *FreqDist()* untuk mendapatkan kata beserta frekuensinya. Setelah itu, didefinisikan fungsi *sort_tuple(tup)*

untuk melakukan *sorting* dari kata yang frekuensinya besar ke rendah, untuk memudahkan analisis.

Untuk *bigram*, yang dapat dilihat dari Gambar 5.4.8 hingga 5.4.15, pertama-tama *bigram* diekstrak dengan cara melakukan *splitting* untuk setiap katanya dan kata[*n*] dan kata[*n+1*] ditambahkan ke kolom *dataframe* baru. Untuk mendapatkan frekuensinya cukup menggunakan fungsi *Counter* dari *Python*.



Gambar 5.4.1 Unigram aspek social

```
print("Top Historical words")
freq_words_list = list(filter(lambda x: x[1]>=5, get_freq_word(text_Historical).items()))
sort_tuple(freq_words_list)
```

```
Top Historical words
[('indonesia', 16),
 ('rakyat', 15),
 ('era', 15),
 ('cinta', 10),
 ('kadrun', 9),
 ('presiden', 9),
 ('ri', 9),
 ('reformasi', 7),
 ('joko', 6),
 ('widodo', 6),
 ('karno', 6),
 ('partai', 6),
 ('demo', 5),
 ('dukung', 5),
 ('cerdas', 5),
 ('sejarah', 5),
 ('megawati', 5),
 ('lupa', 5),
 ('korupsi', 5)]
```

Gambar 5.4.2 Unigram aspek historical

```
print("Top Dehumanization words")
freq_words_list = list(filter(lambda x: x[1]>=5, get_freq_word(text_Dehumanization).items()))
sort_tuple(freq_words_list)
```

```
Top Dehumanization words
[('kadrun', 567),
 ('anjing', 328),
 ('kaing', 228),
 ('comberan', 200),
 ('kakak', 166),
 ('sakit', 146),
 ('indonesia', 143),
 ('presiden', 142),
 ('hati', 139),
 ('ku', 134),
 ('tolol', 130),
 ('ri', 128),
 ('rakyat', 120),
 ('baris', 119),
 ('cinta', 114),
 ('joko', 112),
 ('widodo', 111),
 ('dukung', 98),
 ('wassalam', 98),
 ('gonggong', 98),
```

Gambar 5.4.3 Unigram aspek dehumanization

```
[ ] print("Top Accusation words")
    freq_words_list = list(filter(lambda x: x[1]>=5, get_freq_word(text_Accusation).items()))
    sort_tuple(freq_words_list)

    Top Accusation words
    [('kadrun', 140),
     ('rakyat', 111),
     ('fitnah', 106),
     ('presiden', 101),
     ('hati', 101),
     ('ri', 95),
     ('sakit', 92),
     ('indonesia', 85),
     ('dukung', 82),
     ('joko', 76),
     ('widodo', 76),
     ('benci', 75),
     ('baris', 73),
     ('menang', 72),
     ('sara', 72),
     ('kerja', 71),
     ('teror', 71),
     ('malu', 70),
     ('kakak', 70),
     ('ujar', 67),
     ('bohong', 59),
```

Gambar 5.4.4 Unigram aspek accusation

```
print("Top Attack words")
freq_words_list = list(filter(lambda x: x[1]>=5, get_freq_word(text_Attack).items()))
sort_tuple(freq_words_list)

Top Attack words
[('wanita', 19),
 ('protokol', 11),
 ('sehat', 11),
 ('laksana', 10),
 ('disiplin', 10),
 ('toko', 9),
 ('covid', 6),
 ('malu', 5),
 ('hukum', 5),
 ('pakai', 5)]
```

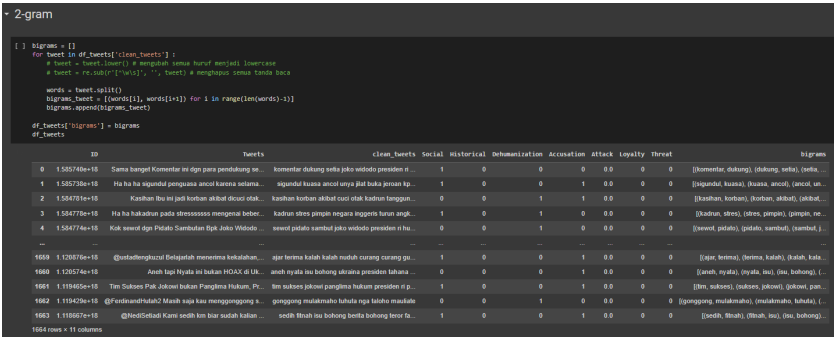
Gambar 5.4.5 Unigram aspek attack

```
[ ] print("Top Loyalty words")
    freq_words_list = list(filter(lambda x: x[1]>=5, get_freq_word(text_Loyalty).items()))
    sort_tuple(freq_words_list)

    Top Loyalty words
    [('dukung', 14),
     ('indonesia', 14),
     ('kadrun', 12),
     ('rakyat', 12),
     ('setia', 10),
     ('pancasila', 10),
     ('kafir', 10),
     ('cinta', 9),
     ('takut', 8),
     ('malu', 8),
     ('joko', 8),
     ('widodo', 8),
     ('presiden', 8),
     ('ri', 8),
     ('khilafah', 8),
     ('ideologi', 7),
     ('ganti', 7),
     ('bela', 5),
     ('ikut', 5),
     ('kakak', 5),
     ('kerja', 5),
     ('negara', 5),
     ('waspada', 5)]
```

Gambar 5.4.6 Unigram aspek loyalty

```
print("Top Threat words")
freq_words_list = list(filter(lambda x: x[1]>=5, get_freq_word(text_Threat).items()))
sort_tuple(freq_words_list)

Top Threat words
[('hukum', 31),
 ('waspada', 30),
 ('tangkap', 21),
 ('said', 20),
 ('didu', 20),
 ('ri', 16),
 ('indonesia', 14),
 ('kadrun', 13),
 ('rakyat', 12),
 ('presiden', 12),
 ('teroris', 12),
 ('joko', 12),
 ('widodo', 12),
 ('paten', 11),
 ('polisi', 11),
```

Gambar 5.4.7 Unigram aspek threat



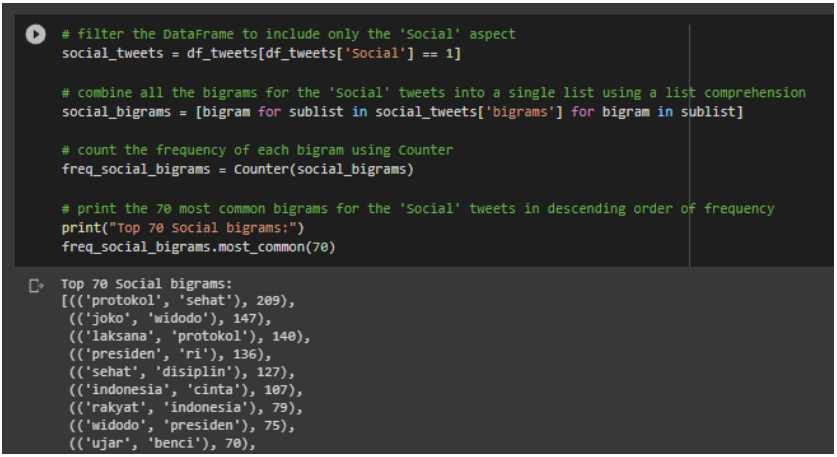
2-gram

```
[ ] bigrams = []
    for tweet in df_tweets['clean_tweets'] :
        # tweet = tweet.lower() # mengubah semua huruf menjadi lowercase
        # tweet = re.sub(r'[^\w\s]', '', tweet) # menghapus semua tanda baca

        words = tweet.split()
        bigrams_tweet = [(words[i], words[i+1]) for i in range(len(words)-1)]
        bigrams.append(bigrams_tweet)

    df_tweets['bigrams'] = bigrams
    df_tweets
```

	ID	Tweets	clean_tweets	Social	Historical	Dehumanization	Accusation	Attack	Loyalty	Threat	bigrams
0	1.585740e+18	Sama banget Komentar ini dgn para pendukung se...	komentar dukung sella joko widodo presiden ri ...	1	0	0	0	0.0	0	0	[(komentar, dukung), (dukung, sella), (sella, ...
1	1.585738e+18	Ha ha ha sigundul penguasa ancol karena selama...	sigundul kuasa ancol unya jilat buka joroan kp...	1	0	0	1	0.0	0	0	[(sigundul, kuasa), (kuasa, ancol), (ancol, un...
2	1.584781e+18	Kasihan Ibu ini jadi korban akibat dicuci otak...	kasihan korban akibat cuci otak kadrun tanggun...	0	0	1	1	0.0	0	0	[(kasihan, korban), (korban, akibat), (akibat,...
3	1.584778e+18	Ha ha hakadrun pada stresssssss mengenai bebec...	kadrun stres pimpin negara inggeris turun angk...	1	0	1	0	0.0	0	0	[(kadrun, stres), (stres, pimpin), (pimpin, ne...
4	1.584774e+18	Kok sewot dgn Pidato Sambutan Bpk Joko Widodo ...	sewot pidato sambut joko widodo presiden ri hu...	0	0	1	0	0.0	0	0	[(sewot, pidato), (pidato, sambut), (sambut, j...
...	...	...	...	...	...	...	...	...	...	...	...
1659	1.120876e+18	@ustadlengkuzul Belajarlah menerima kekalahan,...	ajar terima kalah kalah nuduh curang curang gu...	1	0	0	1	0.0	0	0	[(ajar, terima), (terima, kalah), (kalah, kala...
1660	1.120574e+18	Aneh tapi Nyata ini bukan HOAX di Uk...	aneh nyata isu bohong ukraina presiden tahana ...	0	0	0	1	0.0	0	0	[(aneh, nyata), (nyata, isu), (isu, bohong), (...
1661	1.119465e+18	Tim Sukses Pak Jokowi bukan Panglima Hukum, Pr...	tim sukses jokowi panglima hukum presiden ri p...	1	0	0	1	0.0	0	0	[(tim, sukses), (sukses, jokowi), (jokowi, pan...
1662	1.119429e+18	@FerdinandHutah2 Masih saja kau menggonggong s...	gonggong mulakmaho tuhuta nga laloho maubale	0	0	1	0	0.0	0	0	[(gonggong, mulakmaho), (mulakmaho, tuhuta), (...
1663	1.118667e+18	@NediSeliadi Kami sedih km biar sudah kalian ...	sedih filnah isu bohong berita bohong teror fa...	1	0	0	1	0.0	0	0	[(sedih, filnah), (filnah, isu), (isu, bohong)...

1664 rows × 11 columns

Gambar 5.4.8 Bigram



```
# filter the DataFrame to include only the 'Social' aspect
social_tweets = df_tweets[df_tweets['Social'] == 1]

# combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
social_bigrams = [bigram for sublist in social_tweets['bigrams'] for bigram in sublist]

# count the frequency of each bigram using Counter
freq_social_bigrams = Counter(social_bigrams)

# print the 70 most common bigrams for the 'Social' tweets in descending order of frequency
print("Top 70 Social bigrams:")
freq_social_bigrams.most_common(70)
```

```
Top 70 Social bigrams:
[(('protokol', 'sehat'), 209),
 (('joko', 'widodo'), 147),
 (('laksana', 'protokol'), 140),
 (('presiden', 'ri'), 136),
 (('sehat', 'disiplin'), 127),
 (('indonesia', 'cinta'), 107),
 (('rakyat', 'indonesia'), 79),
 (('widodo', 'presiden'), 75),
 (('ujar', 'benci'), 70),
```

Gambar 5.4.9 Bigram aspek social

```
[ ] # filter the DataFrame to include only the 'Historical' aspect
    historical_tweets = df_tweets[df_tweets['Historical'] == 1]

    # combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
    historical_bigrams = [bigram for sublist in historical_tweets['bigrams'] for bigram in sublist]

    # count the frequency of each bigram using Counter
    freq_historical_bigrams = Counter(historical_bigrams)

    # print the 70 most common bigrams for the 'Historical' tweets in descending order of frequency
    print("Top 70 Historical bigrams:")
    freq_historical_bigrams.most_common(70)

    Top 70 Historical bigrams:
    [(('rakyat', 'indonesia'), 12),
     (('indonesia', 'cinta'), 10),
     (('presiden', 'ri'), 9),
     (('era', 'reformasi'), 6),
     (('joko', 'widodo'), 6),
     (('terima', 'kasih'), 4),
     (('widodo', 'presiden'), 4),
     (('cinta', 'cerdas'), 3),
     (('era', 'orde'), 3),
     (('reformasi', 'rakyat'), 3),
```

Gambar 5.4.10 Bigram aspek historical

```
# filter the DataFrame to include only the 'Dehumanization' aspect
dehumanization_tweets = df_tweets[df_tweets['Dehumanization'] == 1]

# combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
dehumanization_bigrams = [bigram for sublist in dehumanization_tweets['bigrams'] for bigram in sublist]

# count the frequency of each bigram using Counter
freq_dehumanization_bigrams = Counter(dehumanization_bigrams)

# print the 70 most common bigrams for the 'Dehumanization' tweets in descending order of frequency
print("Top 70 Dehumanization bigrams:")
freq_dehumanization_bigrams.most_common(70)

Top 70 Dehumanization bigrams:
[(('sakit', 'hati'), 119),
 (('baris', 'sakit'), 118),
 (('anjing', 'ku'), 118),
 (('presiden', 'ri'), 117),
 (('kaing', 'kaing'), 113),
 (('joko', 'widodo'), 111),
 (('hati', 'kadrun'), 110),
 (('kaing', 'comberan'), 90),
 (('indonesia', 'cinta'), 78),
 (('widodo', 'presiden'), 67),
 (('rakyat', 'indonesia'), 65),
 (('anjing', 'gonggong'), 45),
 (('pesong', 'gila'), 42),
 (('anjing', 'kaing'), 39),
```

Gambar 5.4.11 Bigram aspek dehumanization

```
# filter the DataFrame to include only the 'Accusation' aspect
accusation_tweets = df_tweets[df_tweets['Accusation'] == 1]

# combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
accusation_bigrams = [bigram for sublist in accusation_tweets['bigrams'] for bigram in sublist]

# count the frequency of each bigram using Counter
freq_accusation_bigrams = Counter(accusation_bigrams)

# print the 70 most common bigrams for the 'Accusation' tweets in descending order of frequency
print("Top 70 Accusation bigrams:")
freq_accusation_bigrams.most_common(70)
```

```
Top 70 Accusation bigrams:
[(('presiden', 'ri'), 83),
 (('sakit', 'hati'), 77),
 (('joko', 'widodo'), 76),
 (('baris', 'sakit'), 73),
 (('ujar', 'benci'), 67),
 (('hati', 'kadrun'), 60),
 (('widodo', 'presiden'), 41),
 (('sara', 'ujar'), 41),
 (('fitnah', 'teror'), 40),
 (('indonesia', 'cinta'), 39),
 (('benci', 'fitnah'), 38),
 (('rakyat', 'indonesia'), 35),
```

Gambar 5.4.12 Bigram aspek accusation

```
# filter the DataFrame to include only the 'Attack' aspect
attack_tweets = df_tweets[df_tweets['Attack'] == 1]

# combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
attack_bigrams = [bigram for sublist in attack_tweets['bigrams'] for bigram in sublist]

# count the frequency of each bigram using Counter
freq_attack_bigrams = Counter(attack_bigrams)

# print the 70 most common bigrams for the 'Attack' tweets in descending order of frequency
print("Top 70 Attack bigrams:")
freq_attack_bigrams.most_common(70)
```

```
Top 70 Attack bigrams:
[(('protokol', 'sehat'), 11),
 (('laksana', 'protokol'), 8),
 (('sehat', 'disiplin'), 6),
 (('wanita', 'lemah'), 3),
 (('kena', 'batu'), 3),
 (('gugus', 'tugas'), 3),
 (('tugas', 'covid'), 3),
 (('tokoh', 'nasional'), 3),
 (('kaum', 'wanita'), 2),
 (('he', 'he'), 2),
 (('hukum', 'adat'), 2),
```

Gambar 5.4.13 Bigram aspek attack

```
# filter the DataFrame to include only the 'Loyalty' aspect
loyalty_tweets = df_tweets[df_tweets['Loyalty'] == 1]

# combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
loyalty_bigrams = [bigram for sublist in loyalty_tweets['bigrams'] for bigram in sublist]

# count the frequency of each bigram using Counter
freq_loyalty_bigrams = Counter(loyalty_bigrams)

# print the 70 most common bigrams for the 'Loyalty' tweets in descending order of frequency
print("Top 70 Loyalty bigrams:")
freq_loyalty_bigrams.most_common(70)
```

```
Top 70 Loyalty bigrams:
[(('joko', 'widodo'), 8),
 (('presiden', 'ri'), 8),
 (('indonesia', 'cinta'), 7),
 (('ideologi', 'pancasila'), 6),
 (('widodo', 'presiden'), 5),
 (('ikut', 'setia'), 4),
 (('dukung', 'setia'), 4),
 (('rakyat', 'indonesia'), 3),
 (('ganti', 'ideologi'), 3),
```

Gambar 5.4.14 Bigram aspek loyalty

```
# filter the DataFrame to include only the 'Threat' aspect
threat_tweets = df_tweets[df_tweets['Threat'] == 1]

# combine all the bigrams for the 'Social' tweets into a single list using a list comprehension
threat_bigrams = [bigram for sublist in threat_tweets['bigrams'] for bigram in sublist]

# count the frequency of each bigram using Counter
freq_threat_bigrams = Counter(threat_bigrams)

# print the 70 most common bigrams for the 'Threat' tweets in descending order of frequency
print("Top 70 Threat bigrams:")
freq_threat_bigrams.most_common(70)
```

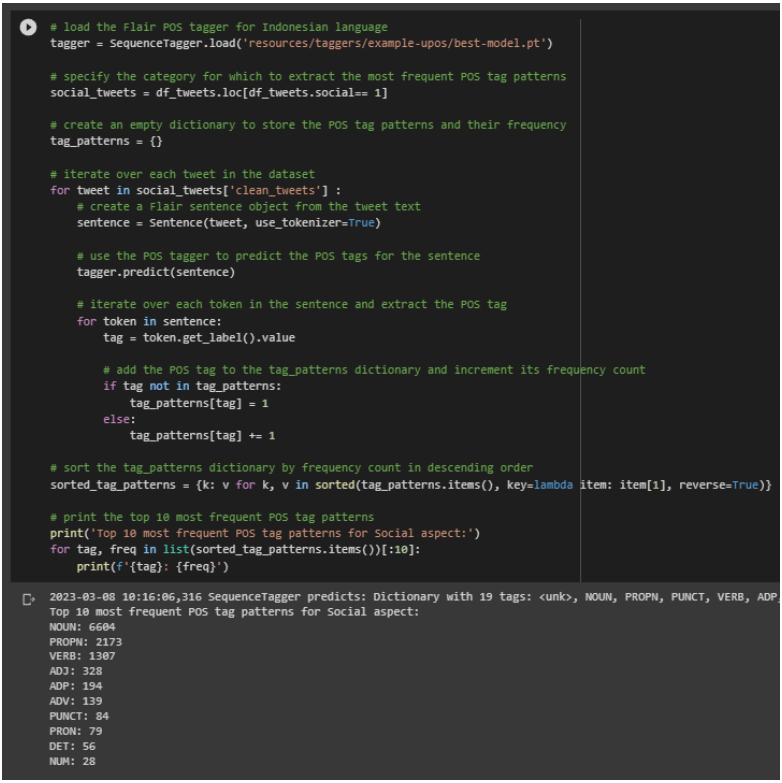
```
Top 70 Threat bigrams:
[(('said', 'didu'), 20),
 (('tangkap', 'said'), 17),
 (('waspada', 'waspada'), 16),
 (('presiden', 'ri'), 12),
 (('joko', 'widodo'), 12),
 (('didu', 'tangkap'), 8),
 (('hukum', 'mati'), 7),
 (('widodo', 'presiden'), 7),
 (('proses', 'hukum'), 5),
```

Gambar 5.4.15 Bigram aspek threat

5.5. Menentukan pola *n-gram* dengan *Part of Speech* (*POS*) *tagging*

Setelah mendapatkan pola *unigram* dan *bigram* dari masing-masing aspek *dangerous speech*, selanjutnya dilakukan penentuan pola dari *unigram* dan *bigram* tersebut, untuk mengetahui pola apa yang sering muncul dari masing-masing aspek. Gambar 5.5.1 hingga 5.5.14 merupakan potongan kode untuk menentukan pola *n-gram*.

Penentuan pola *n-gram* untuk *unigram* dan *bigram* keduanya menggunakan kode yang sama. Pertama-tama *tagger Flair* bahasa Indonesia diinisialisasi, lalu kategori aspek yang ingin diekstrak ditentukan, selanjutnya inisialisasi *dictionary* kosong untuk menyimpan hasil *POS tagging* serta frekuensinya. Selanjutnya untuk setiap *tweet* di *dataset*, dibuat objek *Sentence Flair,* dilakukan prediksi dengan *tagger* yang sudah diinisialisasi, lalu setiap *token* di kalimat diiterasi dan diekstrak pola kalimatnya. Pola kalimat dan frekuensinya akan dimasukkan ke dalam dictionary yang sudah diinisialisasi sebelumnya. Terakhir dilakukan *sorting* dari frekuensi besar ke kecil untuk mempermudah analisis.



```
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# specify the category for which to extract the most frequent POS tag patterns
social_tweets = df_tweets.loc[df_tweets.social== 1]

# create an empty dictionary to store the POS tag patterns and their frequency
tag_patterns = {}

# iterate over each tweet in the dataset
for tweet in social_tweets['clean_tweets'] :
    # create a Flair sentence object from the tweet text
    sentence = Sentence(tweet, use_tokenizer=True)

    # use the POS tagger to predict the POS tags for the sentence
    tagger.predict(sentence)

    # iterate over each token in the sentence and extract the POS tag
    for token in sentence:
        tag = token.get_label().value

        # add the POS tag to the tag_patterns dictionary and increment its frequency count
        if tag not in tag_patterns:
            tag_patterns[tag] = 1
        else:
            tag_patterns[tag] += 1

# sort the tag_patterns dictionary by frequency count in descending order
sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

# print the top 10 most frequent POS tag patterns
print('Top 10 most frequent POS tag patterns for Social aspect:')
for tag, freq in list(sorted_tag_patterns.items())[:10]:
    print(f'{tag}: {freq}')
```

```
2023-03-08 10:16:06,316 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP
Top 10 most frequent POS tag patterns for Social aspect:
NOUN: 6604
PROPN: 2173
VERB: 1307
ADJ: 328
ADP: 194
ADV: 139
PUNCT: 84
PRON: 79
DET: 56
NUM: 28
```

Gambar 5.5.1 Pola unigram aspek social

```python
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# specify the category for which to extract the most frequent POS tag patterns
historical_tweets = df_tweets.loc[df_tweets.historical== 1]

# create an empty dictionary to store the POS tag patterns and their frequency
tag_patterns = {}

# iterate over each tweet in the dataset
for tweet in historical_tweets['clean_tweets'] :
    # create a Flair sentence object from the tweet text
    sentence = Sentence(tweet, use_tokenizer=True)

    # use the POS tagger to predict the POS tags for the sentence
    tagger.predict(sentence)

    # iterate over each token in the sentence and extract the POS tag
    for token in sentence:
        tag = token.get_label().value

        # add the POS tag to the tag_patterns dictionary and increment its frequency count
        if tag not in tag_patterns:
            tag_patterns[tag] = 1
        else:
            tag_patterns[tag] += 1

# sort the tag_patterns dictionary by frequency count in descending order
sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

# print the top 10 most frequent POS tag patterns
print('Top 10 most frequent POS tag patterns for Historical aspect:')
for tag, freq in list(sorted_tag_patterns.items())[:10]:
    print(f'{tag}: {freq}')
```

```
2023-03-08 10:16:19,406 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP
Top 10 most frequent POS tag patterns for Historical aspect:
NOUN: 303
PROPN: 121
VERB: 61
ADJ: 14
ADP: 10
DET: 4
PUNCT: 3
ADV: 3
PRON: 2
```

Gambar 5.5.2 Pola unigram aspek historical

```
[ ]  # load the Flair POS tagger for Indonesian language
     tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

     # specify the category for which to extract the most frequent POS tag patterns
     dehumanization_tweets = df_tweets.loc[df_tweets.dehumanization== 1]

     # create an empty dictionary to store the POS tag patterns and their frequency
     tag_patterns = {}

     # iterate over each tweet in the dataset
     for tweet in dehumanization_tweets['clean_tweets'] :
         # create a Flair sentence object from the tweet text
         sentence = Sentence(tweet, use_tokenizer=True)

         # use the POS tagger to predict the POS tags for the sentence
         tagger.predict(sentence)

         # iterate over each token in the sentence and extract the POS tag
         for token in sentence:
             tag = token.get_label().value

             # add the POS tag to the tag_patterns dictionary and increment its frequency count
             if tag not in tag_patterns:
                 tag_patterns[tag] = 1
             else:
                 tag_patterns[tag] += 1

     # sort the tag_patterns dictionary by frequency count in descending order
     sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

     # print the top 10 most frequent POS tag patterns
     print('Top 10 most frequent POS tag patterns for Dehumanization aspect:')
     for tag, freq in list(sorted_tag_patterns.items())[:10]:
         print(f'{tag}: {freq}')

     2023-03-08 09:58:43,258 SequenceTagger predicts: Dictionary with 20 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP,
     Top 10 most frequent POS tag patterns for Dehumanization aspect:
     PROPN: 4978
     NOUN: 1817
     VERB: 1359
     PUNCT: 500
     ADJ: 390
     PRON: 321
     ADP: 300
     ADV: 266
     AUX: 113
     NUM: 70
```

Gambar 5.5.3 Pola unigram aspek dehumanization

```python
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# specify the category for which to extract the most frequent POS tag patterns
accusation_tweets = df_tweets.loc[df_tweets.accusation== 1]

# create an empty dictionary to store the POS tag patterns and their frequency
tag_patterns = {}

# iterate over each tweet in the dataset
for tweet in accusation_tweets['clean_tweets'] :
    # create a Flair sentence object from the tweet text
    sentence = Sentence(tweet, use_tokenizer=True)

    # use the POS tagger to predict the POS tags for the sentence
    tagger.predict(sentence)

    # iterate over each token in the sentence and extract the POS tag
    for token in sentence:
        tag = token.get_label().value

        # add the POS tag to the tag_patterns dictionary and increment its frequency count
        if tag not in tag_patterns:
            tag_patterns[tag] = 1
        else:
            tag_patterns[tag] += 1

# sort the tag_patterns dictionary by frequency count in descending order
sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

# print the top 10 most frequent POS tag patterns
print('Top 10 most frequent POS tag patterns for Accusation aspect:')
for tag, freq in list(sorted_tag_patterns.items())[:10]:
    print(f'{tag}: {freq}')
```

```
2023-03-08 09:58:58,237 SequenceTagger predicts: Dictionary with 20 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP,
Top 10 most frequent POS tag patterns for Accusation aspect:
PROPN: 3771
NOUN: 1285
VERB: 1168
ADJ: 256
PUNCT: 214
ADP: 189
ADV: 183
PRON: 150
AUX: 112
NUM: 32
```

Gambar 5.5.4 Pola unigram aspek accusation

```python
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# specify the category for which to extract the most frequent POS tag patterns
attack_tweets = df_tweets.loc[df_tweets.attack== 1]

# create an empty dictionary to store the POS tag patterns and their frequency
tag_patterns = {}

# iterate over each tweet in the dataset
for tweet in attack_tweets['clean_tweets'] :
    # create a Flair sentence object from the tweet text
    sentence = Sentence(tweet, use_tokenizer=True)

    # use the POS tagger to predict the POS tags for the sentence
    tagger.predict(sentence)

    # iterate over each token in the sentence and extract the POS tag
    for token in sentence:
        tag = token.get_label().value

        # add the POS tag to the tag_patterns dictionary and increment its frequency count
        if tag not in tag_patterns:
            tag_patterns[tag] = 1
        else:
            tag_patterns[tag] += 1

# sort the tag_patterns dictionary by frequency count in descending order
sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

# print the top 10 most frequent POS tag patterns
print('Top 10 most frequent POS tag patterns for Attack aspect:')
for tag, freq in list(sorted_tag_patterns.items())[:10]:
    print(f'{tag}: {freq}')
```

```
2023-03-08 09:59:09,891 SequenceTagger predicts: Dictionary with 20 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP,
Top 10 most frequent POS tag patterns for Attack aspect:
PROPN: 133
NOUN: 108
VERB: 86
ADJ: 22
PRON: 15
ADV: 14
ADP: 12
AUX: 9
PUNCT: 8
PART: 1
```

Gambar 5.5.5 Pola unigram aspek attack

```
[ ] # load the Flair POS tagger for Indonesian language
    tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

    # specify the category for which to extract the most frequent POS tag patterns
    loyalty_tweets = df_tweets.loc[df_tweets.loyalty== 1]

    # create an empty dictionary to store the POS tag patterns and their frequency
    tag_patterns = {}

    # iterate over each tweet in the dataset
    for tweet in loyalty_tweets['clean_tweets'] :
        # create a Flair sentence object from the tweet text
        sentence = Sentence(tweet, use_tokenizer=True)

        # use the POS tagger to predict the POS tags for the sentence
        tagger.predict(sentence)

        # iterate over each token in the sentence and extract the POS tag
        for token in sentence:
            tag = token.get_label().value

            # add the POS tag to the tag_patterns dictionary and increment its frequency count
            if tag not in tag_patterns:
                tag_patterns[tag] = 1
            else:
                tag_patterns[tag] += 1

    # sort the tag_patterns dictionary by frequency count in descending order
    sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

    # print the top 10 most frequent POS tag patterns
    print('Top 10 most frequent POS tag patterns for Loyalty aspect:')
    for tag, freq in list(sorted_tag_patterns.items())[:10]:
        print(f'{tag}: {freq}')

    2023-03-08 09:59:17,777 SequenceTagger predicts: Dictionary with 20 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP,
    Top 10 most frequent POS tag patterns for Loyalty aspect:
    PROPN: 287
    VERB: 90
    NOUN: 88
    PRON: 17
    AUX: 15
    ADV: 14
    ADP: 12
    PUNCT: 11
    ADJ: 10
    NUM: 5
```

Gambar 5.5.6 Pola unigram aspek loyalty

```
[ ] # load the Flair POS tagger for Indonesian language
    tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

    # specify the category for which to extract the most frequent POS tag patterns
    threat_tweets = df_tweets.loc[df_tweets.threat== 1]

    print(f'Number of tweets in filtered dataframe: {len(df_tweets)}')

    # create an empty dictionary to store the POS tag patterns and their frequency
    tag_patterns = {}

    # iterate over each tweet in the dataset
    for tweet in threat_tweets['clean_tweets'] :
        # create a Flair sentence object from the tweet text
        sentence = Sentence(tweet, use_tokenizer=True)

        # use the POS tagger to predict the POS tags for the sentence
        tagger.predict(sentence)

        # iterate over each token in the sentence and extract the POS tag
        for token in sentence:
            tag = token.get_label().value

            # add the POS tag to the tag_patterns dictionary and increment its frequency count
            if tag not in tag_patterns:
                tag_patterns[tag] = 1
            else:
                tag_patterns[tag] += 1

    # sort the tag_patterns dictionary by frequency count in descending order
    sorted_tag_patterns = {k: v for k, v in sorted(tag_patterns.items(), key=lambda item: item[1], reverse=True)}

    # print the top 10 most frequent POS tag patterns
    print('Top 10 most frequent POS tag patterns for Threat aspect:')
    for tag, freq in list(sorted_tag_patterns.items())[:10]:
        print(f'{tag}: {freq}')

    2023-03-08 10:00:13,245 SequenceTagger predicts: Dictionary with 20 tags: <unk>, NOUN, PROPN, PUNCT, VERB, ADP
    Number of tweets in filtered dataframe: 1664
    Top 10 most frequent POS tag patterns for Threat aspect:
    PROPN: 472
    VERB: 202
    NOUN: 193
    ADJ: 28
    ADP: 27
    PRON: 25
    PUNCT: 20
    AUX: 19
    ADV: 15
    NUM: 6
```

Gambar 5.5.7 Pola unigram aspek threat

2-gram

```
[ ]  # load the Flair POS tagger for Indonesian language
     tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

     # ambil subset data hanya untuk tweet dengan kategori 1
     social_tweets = df_tweets[df_tweets["social"] == 1]["clean_tweets"].tolist()

     # cari pola tag POS bigram paling umum
     patterns = []
     for tweet in social_tweets:
         s = Sentence(tweet)
         tagger.predict(s)
         tags = [token.get_labels()[0].value for token in s]
         bigrams = zip(tags, tags[1:])
         for bigram in bigrams:
             pattern = " ".join(bigram)
             patterns.append(pattern)

     counter = Counter(patterns)
     most_common = counter.most_common(10)

     # mencetak hasil
     print("Top 10 most frequent POS tag bigram patterns in social aspect (Flair, Indonesian model):")
     for i, (pattern, count) in enumerate(most_common):
         print(f"{i+1}. {pattern} ({count} occurrences)")
```

```
2023-03-13 06:25:10,884 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT
Top 10 most frequent POS tag bigram patterns in social aspect (Flair, Indonesian model):
1. NOUN NOUN (2472 occurrences)
2. PROPN PROPN (1663 occurrences)
3. VERB NOUN (1407 occurrences)
4. PROPN VERB (924 occurrences)
5. NOUN PROPN (695 occurrences)
6. NOUN VERB (400 occurrences)
7. NOUN ADJ (340 occurrences)
8. ADJ NOUN (229 occurrences)
9. VERB VERB (181 occurrences)
10. VERB PROPN (163 occurrences)
```

Gambar 5.5.8 Pola bigram aspek social

```python
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# ambil subset data hanya untuk tweet dengan kategori 1
historical_tweets = df_tweets[df_tweets["historical"] == 1]["clean_tweets"].tolist()

# cari pola tag POS bigram paling umum
patterns = []
for tweet in historical_tweets:
    s = Sentence(tweet)
    tagger.predict(s)
    tags = [token.get_labels()[0].value for token in s]
    bigrams = zip(tags, tags[1:])
    for bigram in bigrams:
        pattern = " ".join(bigram)
        patterns.append(pattern)

counter = Counter(patterns)
most_common = counter.most_common(10)

# mencetak hasil
print("Top 10 most frequent POS tag bigram patterns in historical aspect (Flair, Indonesian model):")
for i, (pattern, count) in enumerate(most_common):
    print(f"{i+1}. {pattern} ({count} occurrences)")
```

```
2023-03-13 06:25:22,460 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT, VE
Top 10 most frequent POS tag bigram patterns in historical aspect (Flair, Indonesian model):
1. NOUN NOUN (122 occurrences)
2. PROPN PROPN (88 occurrences)
3. VERB NOUN (63 occurrences)
4. PROPN VERB (49 occurrences)
5. NOUN PROPN (47 occurrences)
6. NOUN VERB (13 occurrences)
7. NOUN ADJ (9 occurrences)
8. VERB VERB (8 occurrences)
9. NOUN PRON (7 occurrences)
10. PROPN PUNCT (7 occurrences)
```

Gambar 5.5.9 Pola bigram aspek historical

```
[ ]  # load the Flair POS tagger for Indonesian language
     tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

     # ambil subset data hanya untuk tweet dengan kategori 1
     dehumanization_tweets = df_tweets[df_tweets["dehumanization"] == 1]["clean_tweets"].tolist()

     # cari pola tag POS bigram paling umum
     patterns = []
     for tweet in dehumanization_tweets:
         s = Sentence(tweet)
         tagger.predict(s)
         tags = [token.get_labels()[0].value for token in s]
         bigrams = zip(tags, tags[1:])
         for bigram in bigrams:
             pattern = " ".join(bigram)
             patterns.append(pattern)

     counter = Counter(patterns)
     most_common = counter.most_common(10)

     # mencetak hasil
     print("Top 10 most frequent POS tag bigram patterns in dehumanization aspect (Flair, Indonesian model):")
     for i, (pattern, count) in enumerate(most_common):
         print(f"{i+1}. {pattern} ({count} occurrences)")

     2023-03-13 06:25:26,741 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT, VERB, A
     Top 10 most frequent POS tag bigram patterns in dehumanization aspect (Flair, Indonesian model):
     1. NOUN NOUN (1987 occurrences)
     2. PROPN PROPN (1490 occurrences)
     3. VERB NOUN (1045 occurrences)
     4. PROPN VERB (686 occurrences)
     5. NOUN PROPN (563 occurrences)
     6. NOUN ADJ (342 occurrences)
     7. NOUN VERB (285 occurrences)
     8. NOUN PRON (256 occurrences)
     9. PROPN PUNCT (212 occurrences)
     10. ADJ NOUN (172 occurrences)
```

Gambar 5.5.10 Pola bigram aspek dehumanization

```python
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# ambil subset data hanya untuk tweet dengan kategori 1
accusation_tweets = df_tweets[df_tweets["accusation"] == 1]["clean_tweets"].tolist()

# cari pola tag POS bigram paling umum
patterns = []
for tweet in accusation_tweets:
    s = Sentence(tweet)
    tagger.predict(s)
    tags = [token.get_labels()[0].value for token in s]
    bigrams = zip(tags, tags[1:])
    for bigram in bigrams:
        pattern = " ".join(bigram)
        patterns.append(pattern)

counter = Counter(patterns)
most_common = counter.most_common(10)

# mencetak hasil
print("Top 10 most frequent POS tag bigram patterns in accusation aspect (Flair, Indonesian model):")
for i, (pattern, count) in enumerate(most_common):
    print(f"{i+1}. {pattern} ({count} occurrences)")
```

```
2023-03-13 06:07:35,187 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT, V
Top 10 most frequent POS tag bigram patterns in accusation aspect (Flair, Indonesian model):
1. NOUN NOUN (3813 occurrences)
2. PROPN PROPN (874 occurrences)
3. NOUN PROPN (348 occurrences)
4. VERB NOUN (269 occurrences)
5. PROPN NOUN (224 occurrences)
6. PROPN VERB (120 occurrences)
7. NOUN ADJ (115 occurrences)
8. PUNCT NOUN (108 occurrences)
9. PROPN PUNCT (100 occurrences)
10. NOUN VERB (90 occurrences)
```

Gambar 5.5.11 Pola bigram aspek accusation

```python
# load the Flair POS tagger for Indonesian language
tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

# ambil subset data hanya untuk tweet dengan kategori 1
attack_tweets = df_tweets[df_tweets["attack"] == 1]["clean_tweets"].tolist()

# cari pola tag POS bigram paling umum
patterns = []
for tweet in attack_tweets:
    s = Sentence(tweet)
    tagger.predict(s)
    tags = [token.get_labels()[0].value for token in s]
    bigrams = zip(tags, tags[1:])
    for bigram in bigrams:
        pattern = " ".join(bigram)
        patterns.append(pattern)

counter = Counter(patterns)
most_common = counter.most_common(10)

# mencetak hasil
print("Top 10 most frequent POS tag bigram patterns in attack aspect (Flair, Indonesian model):")
for i, (pattern, count) in enumerate(most_common):
    print(f"{i+1}. {pattern} ({count} occurrences)")
```

```
2023-03-13 06:07:50,223 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNC
Top 10 most frequent POS tag bigram patterns in attack aspect (Flair, Indonesian model):
1. NOUN NOUN (237 occurrences)
2. PROPN PROPN (24 occurrences)
3. VERB NOUN (15 occurrences)
4. NOUN VERB (11 occurrences)
5. PROPN NOUN (10 occurrences)
6. NOUN PROPN (10 occurrences)
7. NOUN ADJ (10 occurrences)
8. NOUN PRON (7 occurrences)
9. ADJ NOUN (7 occurrences)
10. ADP NOUN (6 occurrences)
```

Gambar 5.5.12 Pola bigram aspek attack

```
[ ]  # load the Flair POS tagger for Indonesian language
     tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

     # ambil subset data hanya untuk tweet dengan kategori 1
     loyalty_tweets = df_tweets[df_tweets["loyalty"] == 1]["clean_tweets"].tolist()

     # cari pola tag POS bigram paling umum
     patterns = []
     for tweet in loyalty_tweets:
         s = Sentence(tweet)
         tagger.predict(s)
         tags = [token.get_labels()[0].value for token in s]
         bigrams = zip(tags, tags[1:])
         for bigram in bigrams:
             pattern = " ".join(bigram)
             patterns.append(pattern)

     counter = Counter(patterns)
     most_common = counter.most_common(10)

     # mencetak hasil
     print("Top 10 most frequent POS tag bigram patterns in loyalty aspect (Flair, Indonesian model):")
     for i, (pattern, count) in enumerate(most_common):
         print(f"{i+1}. {pattern} ({count} occurrences)")

     2023-03-13 06:07:57,044 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT,
     Top 10 most frequent POS tag bigram patterns in loyalty aspect (Flair, Indonesian model):
     1. NOUN NOUN (270 occurrences)
     2. PROPN PROPN (67 occurrences)
     3. NOUN PROPN (25 occurrences)
     4. VERB NOUN (23 occurrences)
     5. PROPN NOUN (17 occurrences)
     6. PROPN VERB (12 occurrences)
     7. NOUN PRON (11 occurrences)
     8. ADP NOUN (7 occurrences)
     9. NOUN VERB (6 occurrences)
     10. NOUN DET (5 occurrences)
```

Gambar 5.5.13 Pola bigram aspek loyalty

```
[ ] # load the Flair POS tagger for Indonesian language
    tagger = SequenceTagger.load('resources/taggers/example-upos/best-model.pt')

    # ambil subset data hanya untuk tweet dengan kategori 1
    threat_tweets = df_tweets[df_tweets["threat"] == 1]["clean_tweets"].tolist()

    # cari pola tag POS bigram paling umum
    patterns = []
    for tweet in threat_tweets:
        s = Sentence(tweet)
        tagger.predict(s)
        tags = [token.get_labels()[0].value for token in s]
        bigrams = zip(tags, tags[1:])
        for bigram in bigrams:
            pattern = " ".join(bigram)
            patterns.append(pattern)

    counter = Counter(patterns)
    most_common = counter.most_common(10)

    # mencetak hasil
    print("Top 10 most frequent POS tag bigram patterns in threat aspect (Flair, Indonesian model):")
    for i, (pattern, count) in enumerate(most_common):
        print(f"{i+1}. {pattern} ({count} occurrences)")

    2023-03-13 06:08:05,735 SequenceTagger predicts: Dictionary with 19 tags: <unk>, NOUN, PROPN, PUNCT,
    Top 10 most frequent POS tag bigram patterns in threat aspect (Flair, Indonesian model):
    1. NOUN NOUN (496 occurrences)
    2. PROPN PROPN (113 occurrences)
    3. NOUN PROPN (60 occurrences)
    4. VERB NOUN (43 occurrences)
    5. PROPN VERB (33 occurrences)
    6. PROPN NOUN (22 occurrences)
    7. NOUN ADJ (17 occurrences)
    8. NOUN VERB (13 occurrences)
    9. ADP NOUN (12 occurrences)
    10. NOUN PRON (11 occurrences)
```

Gambar 5.5.14 Pola bigram aspek threat

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap kamus yang telah dibuat untuk menguji kesesuaian dan ketepatan pembuatan kamus dari data yang telah dilabel.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian kamus didapatkan dengan memperhatikan kesesuaian kata-kata kunci yang terdapat dalam masing-masing aspek.

6.3. Skenario Pengujian

Skenario pengujian melibatkan seorang pembimbing lapangan yaitu Ibu Yulian Findawati S.T., M.MT. untuk melakukan pengecekan kesesuaian kata-kunci untuk setiap aspek dari kamus yang telah dibuat.

6.4. Evaluasi Pengujian

Berdasarkan pengujian yang telah dipaparkan di atas, maka pembimbing lapangan telah menyesuaikan dan menyetujui

kamus yang telah dibuat. Tabel 6.4.1 merupakan rangkuman dari pengujian yang dilakukan.

Tabel 6.4.1 Evaluasi Pengujian

Kriteria Pengujian	**Hasil Pengujian**
Pengumpulan data	Sesuai, hasil *tweet* yang dikumpulkan menggunakan *SNScrape* sesuai dengan parameter yang digunakan.
Pembersihan data	Sesuai, data yang dipra-proses sudah cukup bersih.
Pelabelan *dataset*	Sesuai, pelabelan dataset dilakukan dengan pengertian penulis sendiri mengenai masing-masing aspek *dangerous speech*.
Ekstraksi *n-gram*	Sesuai, *unigram* dan *bigram* berhasil diekstraksi beserta dengan frekuensinya masing-masing.

Penentuan pola kalimat *n-gram*	Sesuai, pola kalimat untuk *unigram* dan *bigram* berhasil diekstraksi beserta dengan frekuensinya.

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melaksanakan kerja praktik adalah sebagai berikut:

1. Proses pengambilan data dilakukan menggunakan *library SNScrape* dan proses pembersihan data mencakup penghapusan *stopword*, penggantian *hashtag* dan *username* yang sering muncul dengan kata aslinya, dan penggunaan lematisasi.
2. Pelabelan data dilakukan untuk keseluruhan ~4000 data *tweet* akun @ruhutsitompul secara manual berdasarkan pengertian *dangerous speech* berdasarkan aspek dari penulis.
3. Ekstraksi *n-gram* serta frekuensi masing-masing *n-gram* dapat dilakukan dengan cara menghitung frekuensi setiap kata untuk setiap aspeknya untuk *unigram*, dan untuk *bigram* dapat dilakukan dengan cara yang sama, namun fitur yang diekstraksi adalah kata dan kata+1 agar didapatkan *bigram*nya.
4. Penentuan pola kalimat *n-gram* yang diperoleh dapat dilakukan dengan menggunakan *POS tagging library flair.*

7.2. Saran

Dalam pelaksanaan kerja praktik terdapat beberapa hal yang masih dapat ditingkatkan:

1. Memperhatikan lebih lagi mengenai pelabelan *dataset* untuk meminimalisir kesalahan pada saat pelabelan
2. Menyatukan format *output* dari kode agar tidak berbeda-beda penulisannya.

DAFTAR PUSTAKA

Benesch, S., Glavinic, T., Manion, S., & Buerger, C. (2018). *Dangerous Speech: A Practical Guide*.

Chiche, A., & Yitagesu, B. (2022). Part of speech tagging: A systematic review of deep learning and machine learning approaches. *Journal of Big Data*, *9*(1), 10. https://doi.org/10.1186/s40537-022-00561-y

JustAnotherArchivist. (2022). *Snscrape* [Python]. https://github.com/JustAnotherArchivist/snscrape (Original work published 2018)

Kurniasih, A., & Manik, L. P. (2022). On the Role of Text Preprocessing in BERT Embedding-based DNNs for Classifying Informal Texts. *International Journal of Advanced Computer Science and Applications*, *13*(6). https://doi.org/10.14569/IJACSA.2022.01306109

Statista. (2022). *Countries with most Twitter users 2022*. Statista. https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/

Zhu, L., Wang, W., Huang, M., Chen, M., Wang, Y., & Cai, Z. (2022). A N-gram based approach to auto-extracting topics from research articles1. *Journal of Intelligent & Fuzzy*

Systems, *43*(5), 6137–6146. https://doi.org/10.3233/JIFS-220115

BIODATA PENULIS

Nama	: Christian Bennett Robin
Tempat, Tanggal Lahir	: Jakarta, 26 November 2001
Jenis Kelamin	: Laki-laki
Agama	: Kristen
Status	: Belum Menikah
Alamat	: Jl. Jembatan Besi Jaya No.10
Telepon	: 087726112001
Email	: christianbennettrobin@gmail.com

PENDIDIKAN FORMAL

2019 – sekarang	: S1 Teknik Informatika ITS
2016 – 2019	: SMAK 1 PENABUR Jakarta
2013 – 2016	: SMPK 2 PENABUR Jakarta
2008 – 2013	: SDK Pelangi Kasih

AKADEMIS

Kuliah : Departemen Teknik Informatika – FTEIC ITS

Angkatan : 2019

Semester : 8 (Delapan)

IPK : 3.59