Implementasi *Deep Learning* Untuk Mengubah Kalimat Tidak Sopan Menjadi Sopan

Disusun Oleh:

N a m a : Axel Christiant Kusuma

NIM : 17523232

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

Implementasi *Deep Learning* Untuk Mengubah Kalimat Tidak Sopan Menjadi Sopan

TUGAS AKHIR

Disusun Oleh:

N a m a : Axel Christiant Kusuma

NIM : 17523232

Yogyakarta, 13 Januari 2022

Pembimbing,

(Ahmad M. Raf'ie Pratama, S.T, M.IT, Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

Implementasi *Deep Learning* Untuk Mengubah Kalimat Tidak Sopan Menjadi Sopan

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 13 Januari 2022

Tim Penguji

Ahmad M. Raf'ie Pratama, S.T, M.IT, Ph.D. _____

Anggota 1

Chanifah Indah Ratnasari, S.KOM, M.KOM. _____

Anggota 2

Rian Adam Rajagede, S.KOM, M.CS. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Axel Christiant Kusuma

NIM : 17523232

Tugas akhir dengan judul:

Implementasi *Deep Learning* Untuk Mengubah Kalimat Tidak Sopan Menjadi Sopan

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 13 Januari 2022

(Axel Christiant Kusuma)

HALAMAN PERSEMBAHAN

Alhamdulillahirobbil'alamin atas segala nikmat yang telah diberikan kepada kita semua oleh Allah subhanahu wa ta'ala. Shalawat serta salam kita haturkan kepada junjungan kita Nabi Muhammad SAW yang kita nantikan safa'atnya di yaumul akhir nanti.

Saya ingin berterimakasih sebesar-besarnya kepada orang tua saya yang telah mendidik, merawat, dan mengasihi saya dari lahir. Saya juga ingin mengucapkan terimakasih kepada tante saya yang telah membiayai kuliah saya ketika kedua orang tua saya meninggal dan telah membantu saya untuk melewati kesulitan dalam hidup sampai sekarang.

Terima kasih kepada dosen pembimbing saya, Bapak Ahmad Munasir Raf'ie Pratama, ST., M.I.T., Ph.D. yang membantu saya dalam menyelesaikan tugas akhir untuk Mendapatkan gelar sarjana. Terimakasih juga kepada seluruh dosen-dosen Informatika yang Telah Memberikan ilmu yang semoga nantinya akan berguna bagi masyarakat dan negara.Semoga Allah SWT kelak akan membalas kebaikan kalian semua di akhirat maupun dunia, amin.

HALAMAN MOTO

"*Waste no more time arguing about what a good man should be.*
Be one"

Marcus Aurelius

"*The great teacher, Failure is''*

Yoda

KATA PENGANTAR

Alhamdulillah puji dan syukur kepada Allah *SWT* atas segala nikmat, rahmat, dan ridhonya sehingga penulis dapat menyelesaikan laporan tugas akhir ini. Shalawat beriring salam semoga senantiasa terlimpah dan tercurahkan kepada junjungan kita nabi besar yaitu Nabi Muhammad *SAW*, beserta keluarga dan para sahabatnya, hingga kepada umatnya di akhir zaman, amin.

Tugas akhir berjudul "Implementasi *deep learning* untuk mengubah kalimat tidak sopan menjadi sopan" bertujuan untuk memenuhi persyaratan yang harus dipenuhi untuk menyelesaikan Pendidikan pada jenjang Strata 1 di Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Pada kesempatan kali ini saya ingin menyampaikan rasa terimakasih yang sebesar-besarnya kepada:

1. Orang tua dan keluarga yang telah mendukung perkembangan saya dan selalu membantu saya.
2. Bapak Hendrik, ST., M.Eng. selaku Ketua Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Program Studi Informatika Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Ahmad Munasir Raf'ie Pratama, ST., M.I.T., Ph.D. selaku Dosen Pembimbing Skripsi, atas segala bantuan, bimbingan, dan dukungannya.
5. Seluruh dosen Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia atas ilmu yang nantinnya akan berguna bagi bangsa dan negara.
6. Teman–teman HBS yang selalu membantu saya.

Saya sadar banyak terdapat kekurangan dalam pembuatan tugas akhir ini. Meskipun laporan tugas akhir ini jauh dari kata sempurna, semoga dapat menjadi pelajaran dari sisi baik maupun buruknya bagi penelitian–penelitian terkait yang selanjutnya.

Yogyakarta, 13 Januari 2022

(Axel Christiant)

SARI

Menurut survei yang dilakukan oleh *Microsoft* pada tahun 2020, Indonesia menjadi negara dengan kesopanan digital paling buruk di Asia Pasifik. Hal tersebut dibuktikan dengan naiknya angka *Digital Civility Index* delapan poin dari tahun 2019 menjadi 76 poin. Hal tersebut mendorong penelitian ini untuk membangun model *deep learning* yang dapat mengubah kalimat tidak sopan menjadi sopan dengan strategi mempertahankan bentuk formal pada setiap katanya. Tujuan dari penelitian ini adalah membantu masyarakat Indonesia untuk tetap dapat berkomunikasi dengan sopan di dunia digital sehingga diharapkan dapat menjaga keharmonisan pada saat berkomunikasi dan memperbaiki citra masyarakat Indonesia di dunia digital. Metode yang digunakan penelitian ini adalah *Tag and Generate Approach*; model *tagger* untuk menggantikan *token tag* pada kata yang terdapat di kalimat tidak sopan dan model *generator* untuk menggantikan *token tag* tersebut dengan kata yang sesuai sehingga menjadi kalimat yang sopan. Sebelum melakukan pelatihan model *tagger*, setiap *n-gram* (penelitian ini menggunakan jangkauan *unigram* sampai *bigram*) dilakukan penghitungan rasio rerata *tf-idf* untuk mengetahui peringkat persentil dan relevansi *n-gram* pada masing-masing gaya teks. Dapat disimpulkan bahwa kata "mengapa" dan "kalo" menduduki peringkat persentil *unigram* tertinggi, sedangkan kata "bagaimana ini" dan "ya min" mendapatkan peringkat persentil tertinggi *bigram* pada masing-masing gaya teks. Hasil akhir evaluasi model mencapai nilai tertinggi pada BLEU 1 dengan nilai 0.605 dan disusul oleh METEOR sebesar 0.573. Sedangkan, untuk metrik BLEU 2, BLEU 3, dan BLEU 4 masih tertinggal jauh dibandingkan kedua metrik tersebut (BLEU 1 dan METEOR) dengan nilai masing-masing 0.475, 0.385, dan 0.318.

Kata kunci: *Text Style Transfer*, *Politeness*, *Tag and Generate*, *Deep Learning*.

GLOSARIUM

AMT	*Amazon mechanical Turk* adalah jasa *Crowd Sourcing* yang ditawarkan oleh perusahaan *Amazon*.
Dataset	Kumpulan data yang digunakan untuk pelatihan model *deep learning*.
Deep Learning	Metode pembelajaran oleh mesin dengan cara meniru bagaimana otak manusia bekerja.
Korpus	Kumpulan data teks autentik dalam jumlah besar yang disimpan secara elektronik.
Model	Representasi atau deskripsi dari sistem *deep learning*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kesopanan adalah perilaku yang mempertimbangkan perasaan orang lain tentang bagaimana mereka harus diperlakukan secara interaksional yang menunjukkan kepedulian terhadap status dan hubungan sosial dari para pelaku interaksi (Brown, 2015). Definisi tersebut menunjukkan bahwa kesopanan merupakan salah satu hal yang krusial dalam membangun dan mempertahankan hubungan sosial antar manusia.

Kemajuan teknologi membuat manusia semakin mudah dalam melakukan komunikasi. Namun, di sisi lain hal ini juga menyebabkan manusia menjadi merasa bebas dari norma kesopanan karena tidak adanya pengawasan langsung dari pihak otoritas. Hal tersebut dapat dibuktikan dengan survey yang dilakukan oleh *Microsoft* yang mempunyai parameter bernama *DCI* atau *Digital Civility Index* yaitu merupakan indeks yang mengukur tingkat adab atau kesopanan masyarakat di setiap negara saat berselancar di dunia digital. *DCI* mempunyai skala dari 0 sampai 100 poin yang semakin tinggi nilainya maka semakin buruk tingkat kesopanan masyarakat di negara tersebut. Indonesia menjadi negara dengan kesopanan digital paling buruk di Asia Pasifik, dengan *DCI* bernilai 76 pada tahun 2020 yang naik delapan poin dari tahun 2019 (Yosepha Pusparisa, 2021). Nilai tersebut menunjukkan bahwa masyarakat Indonesia belum dapat melaksanakan norma kesopanan dengan baik di dunia digital terutama masyarakat dengan usia dewasa dan remaja dengan kontribusi 83% dan 68%.

Ketidaksantunan dalam berbahasa terjadi karena adanya penggunaan tuturan yang informal dalam situasi yang formal (adanya jarak sosial) atau sebaliknya (Pranowo, 2012). Oleh karena itu, konteks perlu digunakan dalam memahami dan menghasilkan tuturan untuk membangun kerjasama dan sopan santun dalam proses komunikasi sehingga tujuan komunikasi dapat dicapai secara efektif (Abdurrahman, 2006). Meskipun banyak orang menyadari pentingnya kesantunan berbahasa dalam komunikasi bukan berarti hal ini mudah dilakukan di dunia digital. Hal ini dapat disebabkan karena kurang kritisnya seseorang akan tuturan dalam berkomunikasi, kurangnya pengetahuan dalam kesantunan berbahasa, dan butuhnya menyampaikan tuturan secara cepat dalam berkomunikasi sehingga tidak sempat mengubahnya menjadi bahasa yang sopan. Oleh karena itu, dalam berkomunikasi di media sosial dibutuhkanlah sistem yang dapat membantu seseorang untuk berkomunikasi secara sopan.

Penelitian ini akan membangun sistem *NLG (Natural Language* Generation*)* berjenis *Text*

Style Transfer Transfer untuk mengubah kalimat tidak sopan menjadi kalimat sopan dengan metode *Tag and* Generate yang bersumber pada penelitian sebelumnya yang berjudul *Politeness Transfer : A Tag and Generate Approach* (Madaan et al., 2020), yaitu penelitian yang membangun model *TST* dengan metode *Tag and Generate* menggunakan data *email* karyawan *Enron.* Hal tersebut sejalan dengan tujuan penelitian ini yaitu membantu masyarakat Indonesia untuk tetap mempertahankan adab kesantuan berbahasa di dunia digital khususnya sosial media sehingga dapat menurunkan angka *DCI* negara Indonesia ke depannya.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat disimpulkan rumusan masalah dalam penelitian ini adalah:

a. Bagaimana cara mengubah kalimat tidak sopan menjadi sopan dalam bahasa Indonesia dengan menggunakan model *Tag and Generate?*
b. Apakah model *Tag and Generate* dapat memberikan hasil yang baik untuk mengubah kalimat tidak sopan menjadi sopan dalam bahasa Indonesia?
c. Apa hasil analisis akhir model *Tag and Generate* untuk mengubah kalimat tidak sopan menjadi sopan dalam bahasa Indonesia?

1.3 Batasan Masalah

Penelitian ini terdapat batasan masalah untuk membantu penelitian ini lebih terarah dan sesuai dengan tujuan yang dimaksud, batasan masalah pada penelitian ini adalah sebagai berikut:

a. Dataset yang digunakan merupakan hasil modifikasi dataset dari penelitian yang berjudul *Semi-Supervised Low-Resource Style Transfer of Indonesian Informal to Formal Language with Iterative Forward-Translation* (2020).
b. Dataset yang digunakan merupakan hasil dari *scrapping* melalui akun *Twitter* yang memiliki domain *Customer Service.*
c. Kesopanan yang terdapat pada dataset merupakan strategi kesopanan dalam berkomunikasi dengan menggunakan bentuk kalimat berbentuk formal (Kardana et al., 2018).

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk membantu masyarakat Indonesia untuk mempertahankan adab kesantuan berbahasa di dunia digital sehingga dapat menurunkan angka *DCI* dan

memperbaiki citra negara Indonesia. Hal tersebut diwujudkan pada penelitian ini dengan cara membuat model *deep learning* yang dapat mengubah kalimat tidak sopan menjadi sopan dengan strategi mempertahankan bentuk formal pada setiap katanya.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan kontribusi dan dorongan untuk penelitian selanjutnya untuk mengembangkan model yang lebih baik sehingga dapat diimplementasikan di dunia nyata dengan menjadikannya sebagai sistem pengubah kalimat menjadi lebih sopan. Dengan begitu, hasil dari penelitian ini dan berikutnya diharapkan dapat membantu menurunkan angka *Digital Civility Index* dan memperbaiki citra kesopanan digital negara Indonesia.

BAB II
KAJIAN PUSTAKA

2.1 Text Style Transfer

Gaya teks (*Text Style*) adalah variasi linguistik pada teks dengan tetap mempertahankan konteks yang ingin disampaikan. Dalam melakukan komunikasi, gaya teks yang digunakan setiap orang merupakan sesuatu yang situasional dan tidak pasti. Setiap kata penyusunnya didasarkan pada waktu, tempat, dan skenario tertentu untuk menyampaikan maksud dari pembicara dengan karakteristiknya sendiri (Jin et al., 2020). Secara intuitif, gaya pada teks mendeskripsikan tentang bagaimana cara komunikator memilih kata yang dipakai dan menyusun kalimat yang dipakai untuk membentuk nada, gambar, dan semantik pada teks (Hu et al., 2020).

Beberapa tahun terakhir, penelitian tentang gaya pada teks atau *Text Style* menarik perhatian tidak hanya ahli bahasa namun juga ilmuwan komputer. Secara spesifik, penelitian yang dilakukan oleh ilmuwan komputer adalah transfer gaya pada kalimat atau disebut juga dengan *Text Style Transfer (TST). Text Style Transfer* adalah bagian cabang dari *Natural Language Generation (NLG)* yang bertujuan untuk mengubah suatu gaya pada suatu teks dengan tetap mempertahankan konten pada teks tersebut. Pada awal perkembangannya, *TST* masih berkaitan erat dengan *Neural Machine Translation (NMT)* dan *Neural Style Transfer (NST).* Hal tersebut dikarenakan *NMT* dan *NST* juga merupakan suatu cabang dari *Natural Language Generation* yang bertujuan untuk melatih komputer menghasilkan teks yang dapat dipahami oleh manusia berdasarkan *input* dari data yang ada.

Untuk menggambarkan suatu implementasi dari *Text Style Transfer,* diasumsikan terdapat dua jenis teks yaitu: formal (x) dan informal (x'). Tugas sebuah model *TST* adalah menerima *input* x dengan atribut dasar teks s dan model tersebut akan menghasilkan *output* x' menggunakan attribut teks pada target t dengan tetap mempertahankan konten dan konteks pada x (Hu et al., 2020).

Terdapat dua jenis data yang digunakan untuk melatih model *Text Style Transfer* yaitu paralel dan non-paralel. Data berjenis paralel berisi sepasang teks yang masing-masing memiliki konteks yang sama namun gaya teks yang berbeda. Berlawanan dengan paralel, data berjenis non-paralel mempunyai sepasang teks yang berbeda konteks dan gaya teks. Dataset berjenis paralel membutuhkan banyak pekerjaan manusia dalam membuat gaya teks tujuannya, karena setiap pasangan kalimat dengan arti yang sama namun cara penyampaian kedua kalimat tersebut berbeda. Dataset yang tersedia saat ini merupakan data hasil dari penelitian *TST*

sebelumnya dengan gaya teks asal dan tujuan masing-masing, seperti penelitian yang dilakukan oleh Sudha Rao dan Joel Tetreault untuk mengubah kalimat informal menjadi formal dan sebaliknya dengan membangun dataset paralel *GYAFC (Grammarly's Yahoo Answer Formality Corpus)* dengan mengambil dari *Yahoo Answer L6 Corpus* yaitu kumpulan jawaban dari *Yahoo answer*. *GYAFC* berisikan 112.975 pasangan teks informal ke formal dan 111.266 pasangan teks formal dan informal yang dalam setiap pekerjaan manual merubah ke teks formalnya dibantu oleh *Amazon Mechanical Turk (AMT)* dan beberapa ahli di bidang bahasa (Rao & Tetreault, 2018). Penelitian dengan tujuan terkait yang dilakukan oleh Wibowo dkk menggunakan data paralel yang diambil dari platform media sosial *twitter* tetapi hanya fokus pada domain *Customer Service* berjumlah 2500 pasangan teks informal ke formal.

Dalam pengimplementasiannya di dunia nyata, *Text Style Transfer* dapat berguna untuk membantu seseorang dalam melakukan pekerjaan yang melibatkan penyusunan kata pada teks seperti:

1. Membantu kegiatan menulis untuk siswa yang mempunyai *learning disabilities* dengan menggunakan metode *Word Processing, Spelling Checkers, Word Prediction,* dan *Speech Recognition* untuk memperbaiki struktur pada teks yang ditulis (MacArthur, 2009).
2. Mengubah kalimat yang diucapkan seorang ahli agar dapat dipahami oleh orang awam (Cao et al., 2020).
3. Menurut penelitian yang dilakukan oleh Barbara, gaya pada suatu teks memiliki efek terhadap tingkat persuasif dari apa yang disampaikan (Johnstone, 1989). Dengan begitu, TST dapat membantu manusia pada bidang marketing untuk mengubah kalimat menjadi lebih persuasif.
4. Membuat *Chatbot* menjadi lebih interaktif sehingga percakapan yang dihasilkan dengan *user* akan lebih menarik dan persuasif.

Untuk melakukan evaluasi pada model *Text Style Transfer* terdapat tiga objektif untuk mengukur kualitas suatu model:

1. ***Content Preservation*** atau kelestarian konten pada teks yang dihasilkan oleh model tidak berubah terhadap *input*. **BLEU** (Papineni et al., 2002a) dan **METEOR** (Lavie & Agarwal, 2007) adalah metrik yang dibuat untuk mengukur kualitas dari *Machine Translation* atau sistem penerjemah bahasa. Kedua metrik tersebut menghitung skor berdasarkan kesamaan antara teks yang dihasilkan oleh model dan referensi teks yang dibuat untuk evaluasi model. **BLEU** mengukur teks pada *output* dengan menghitung

kesamaan pada *n—grams* atau jumlah kata yang sama pada kedua teks (*output* dan teks *reference*) menggunakan *precision* (Madaan et al., 2020) sedangkan **METEOR** adalah metrik hasil dari modifikasi *weighted F-Score* berdasarkan pencocokan *unigram* dan *penalty function* untuk kata yang tidak sesuai urutan.

2. ***Quality of style*** atau kualitas model untuk menghasilkan gaya teks target pada *output.* Untuk mengukur kualitas gaya teks yang dihasilkan oleh model diperlukannya *classifier* untuk menghitung persentase akurasi teks tersebut terhadap gaya teks yang dituju. Kebanyakan pada penelitian *Text Style Transfer* sebelumnya (Li et al., 2018) (Dos Santos et al., 2018) (Zhang et al., 2020) menggunakan *pre-trained classifier* untuk mengukur akurasi keakuratan gaya teks model mereka. Semakin tinggi akurasi, kualitas gaya teks yang dihasilkan model. Metrik berupa *precision, recall, dan F1* score juga dapat mengukur kualitas model dalam hal tersebut.
3. ***Fluency*** atau kefasihan bahasa pada teks yang dihasilkan oleh model untuk dapat dipahami oleh manusia secara kontekstual. *The Kneser-Ney language model* (Kneser & Ney, 1995) adalah *pre-trained model* yang biasanya digunakan untuk mengukur kualitas kefasihan bahasa pada model. *The Kneser-Ney language model* mengukur *perplexity score* dari teks yang dihasilkan oleh model dengan cara membandingkan *trigram* pada teks dengan distribusi *trigram* yang telah diestimasi pada pelatihan model. Semakin kecil *perplexity score* yang didapat maka semakin fasih model *TST* tersebut (Hu et al., 2020).

2.2 Metode *Tag and Generate Approach*

Metode *Tag and Generate Approach* pada *Text Style Transfer* dibuat oleh Aman Madaan (Madaan et al., 2020) dengan tujuan untuk mengubah kalimat yang tidak sopan menjadi kalimat yang sopan menggunakan data email karyawan *Enron.* Metode tersebut dibagi menjadi dua tahap, yaitu:

1. ***Tagger*:** Mengidentifikasi kata atau frasa yang merupakan suatu atribut pada gaya teks tersebut lalu mengubahnya menjadi *token tag*.
2. ***Generator*:** Menerima *input* dari *tagger* dan mengisi *token tag* dengan atribut pada gaya teks target. Jika suatu *input* sudah tidak memiliki *token tag* maka teks tersebut sudah memenuhi atribut pada gaya teks target dan tahap ini menghasilkan teks yang sama dengan *input*.

Gambar 2.1 Implementasi pipeline *Tagger* dan *Generator*
Sumber: *Madaan et al.* (2020)

Gambar 2.1 merupakan implementasi *pipeline Tag and Generate approach*. Pada contoh pertama (***add-tagger***), teks $\mathbf{x_1^{(1)}}$ tidak mempunyai atribut gaya teks yang menjadi karateristik teks tersebut, dengan begitu *tagger* menambahkan *token tag* tanpa harus menghapus kata atau frasa yang terdapat pada teks z(x1). Pada contoh kedua (***replace-tagger***), teks $x_2^{(1)}$ mempunyai atribut teks berupa kata *ok* dan *bland* yang menjadi karateristik gaya teks *sentiment negative* karenanya *tagger* menghapus dan mengisi kata tersebut dengan *token tag* pada teks z(x2). Setelah teks diproses oleh tahap *tagger*, *output* pada model tersebut menjadi *input* untuk model *generator* menggantikan *token tag* menjadi kata atau frasa yang memiliki karakteristik gaya teks pada target. Hasil yang didapatkan pada metode ini jika dibandingkan dengan metode penelitian sebelumnya yaitu *Delete, Retrieve, and Generate* (Li et al., 2018). dengan dataset yang sama; mengalami peningkatan sebesar 58.61 pada metrik BLEU dan 18.07 pada metrik METEOR

2.3 Strategi Kesopanan Dalam Bahasa Indonesia

Bahasa merupakan salah satu alat yang paling penting dalam kegiatan komunikasi antar manusia, tidak hanya sebagai perantara untuk menyampaikan sesuatu namun juga sebagai alat integrasi dan adaptasi terhadap situasi dan lingkungan sosial yang sedang dihadapi oleh pelaku komunikasi. Dengan begitu, seseorang dapat menggunakan bahasa yang berbeda satu sama lain ketika berkomunikasi dengan orang yang berbeda. Sebagai contohnya, seseorang yang berasal dari Yogyakarta menggunakan bahasa Jawa ngoko (kasar) saat berkomunikasi dengan orang yang sepantaran atau lebih muda dan menggunakan Jawa krama (halus) ketika berbicara dengan seseorang yang lebih tua.

Menurut Jumanto (2014), bahasa harus dapat diarahkan pada kesopanan (*Distant language*) dan kedekatan (*Close Language*) (Faculty et al., 2014). *Distant Language* dan *Close Language* merujuk pada jarak sosial (*Social Distance*) antara pendengar dan pembicara. *Social Distance* adalah suatu jarak yang memisahkan kedua individu berdasarkan sisi jasmani ataupun psikologi. *Distant Language* digunakan ketika terdapat jarak sosial pada kedua pelaku komunikasi seperti umur, status sosial, ataupun kedekatan hubungan antar individu, sedangkan *Close language* digunakan pada situasi sebaliknya. Dalam bahasa Indonesia, ketidaksopanan terjadi ketika penggunaan *Distant* dan *Close language* tersebut tidak sesuai dengan aturan yang berlaku. Oleh karena itu, setiap individu perlu memperhatikan kesopanan dalam berbahasa (*Distant Language*) pada situasi yang formal ataupun berhadapan dengan seseorang yang patut dihormati.

Menurut penelitian yang dilakukan oleh Kardana (2018), terdapat lima strategi yang dapat digunakan untuk mengekspresikan kesopanan dalam bahasa Indonesia, yaitu:

1. **Pengunaan kalimat tidak langsung**

   Kalimat tidak langsung adalah kalimat yang maknannya tidak sesuai dengan apa yang disampaikan atau terdapat makna tersembunyi yang sebenarnya daripada yang disampaikan. Contoh:

   1. Wah, panas sekali ya ruangannya.
   2. Bisakah anda menyelesaikan pekerjaan itu dalam tiga hari?

   Kalimat pertama merupakan sebuah kalimat deklaratif yang tujuannya adalah menyampaikan informasi kepada pendengar. Namun, dibalik kalimat tersebut terdapat maksud tersembunyi dari pembicara kepada pendengar untuk inisiatif menyalakan pendingin ruangan. Sedangkan, kalimat kedua adalah kalimat tanya yang makna dibaliknya terdapat pernyataan bahwa pembicara meminta pendengar untuk menyelesaikan tugasnya dalam waktu tiga hari. Kalimat pertama dan kedua lebih dianggap sopan jika dibandingkan dengan kalimat imperatif seperti:

   1. Nyalakan pendingin ruangannya!
   2. Selesaikan pekerjaan itu dalam tiga hari!

2. **Pengunaan deiksis persona**

   Deiksis persona merupakan pronomina persona yang bersifat ekstratekstual yang berfungsi menggantikan suatu acuan (anteseden) di luar wacana (Sudaryat, 2009). Deiksis persona menjadi salah satu dari beberapa leksikon yang dibutuhkan untuk

menunjukkan kesopanan dalam bahasa Indonesia. Secara sederhananya, deiksis persona digunakan untuk kata ganti pembicara yang menunjukkan adanya jarak sosial (*Social distance*) dengan pendengar. Contoh:

1. Ya Tuhan, ampunilah **hamba**mu ini
2. Maaf pak, **Bapak** ada di rumah besok?
3. **Ibu** ada acara siang ini?

Kalimat pertama menunjukkan bahwa pembicara tidak berdaya atau mempunyai kekuatan yang jauh tidak sebanding dengan pendengar (Tuhan). Kalimat kedua dan ketiga menunjukkan adanya jarak usia antara pembicara dan pendengar. Dari ketiga kalimat tersebut pembicara menggantikan kata anda atau kamu yang merujuk pada pendengar sesuai dengan jarak sosial yang ada.

3. Penggunaan kata ganti yang tepat

Ketika berkomunikasi dengan anggota keluarga, kerabat, atau kolega sehari-hari, kata ganti yang tepat dibutuhkan untuk memenuhi kesopanan dalam bahasa Indonesia. Contoh:

1. Bu, **Diva** pulang telat ya. Masih ada kegiatan lain di sekolah.
2. **Komang** ikut Ibu keluar, mau beli buku.

Penggunaan kata Diva dan Komang di atas menggantikan kata “aku” dan “saya” yang dianggap kurang sopan penggunaanya jika dipakai untuk berkomunikasi dengan kerabat keluarga.

4. Penggunaan bentuk formal atau baku

Strategi ini tepat jika digunakan ketika pembicara dalam situasi tertentu seperti hadir di acara formal atau resmi, bertemu dengan orang yang statusnya lebih tinggi, atau berbicara dengan orang yang baru dikenal. Berikut ini adalah contoh kalimat informal yang diubah menjadi formal:

1. **Informal**:

   Pada kesempatan ini saya ingin **ngomongin** tentang program kita bulan ini.

   Formal:

   Pada kesempatan ini saya ingin **mengatakan** tentang program kita bulan ini.
2. **Informal:**

   Yang saya ingin **omongkan** adalah masalah yang berkaitan dengan kakak anda.

Formal:

Yang saya ingin **katakan** adalah masalah yang berkaitan dengan kakak anda.

3. **Informal:**

   Hal itu **nggak usah** dibahas lagi.

   Formal:

   Hal itu **tidak perlu** dipermasalahkan lagi.

4. **Informal:**

   Yang **cewek-cewek** duduk di sebelah sana dan **cowok-cowok** di sebelah sini.

   Formal:

   Yang **perempuan** agar duduk di sebelah sana dan **laki-laki** di sebelah sini.

   Keempat kalimat di atas menunjukkan perubahan beberapa kata yang tidak baku (informal) menjadi kata yang baku (formal) sesuai dengan pedoman dan kaidah bahasa Indonesia. Perubahan tersebut membuat kalimat lebih baik dalam aturan tata bahasa dan membuat komunikasi menjadi lebih baik dan sopan di situasi tertentu.

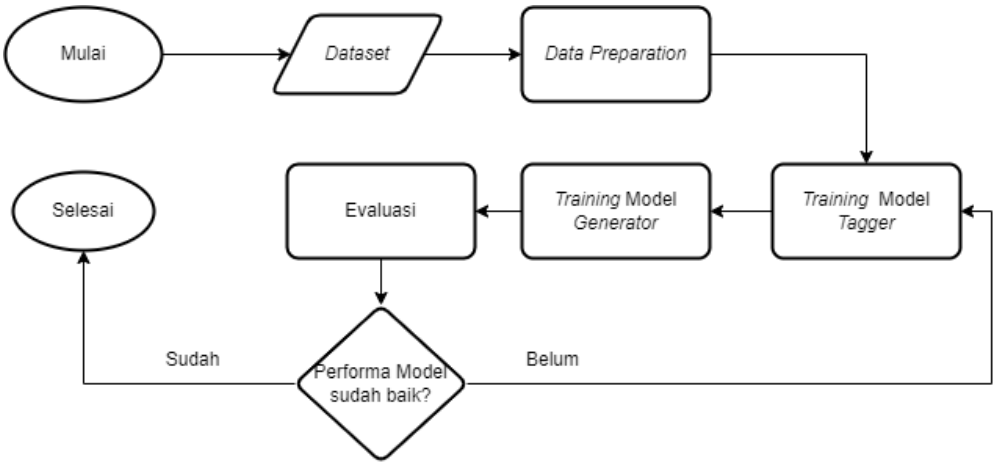
5. Penggunaan kata kerja pasif

Kata kerja pasif dapat digunakan untuk mengekspresikan kesopanan dalam bahasa Indonesia. Kata tersebut biasanya digunakan oleh *MC* (*Master of Ceremony*) untuk membuka acara yang formal, seperti contohnya:

1. Hadirin **dimohon** berdiri.
2. Pak ketua **disilahkan** untuk menempati tempat yang sudah disediakan.

Kata “dimohon” dan “disilahkan” merupakan kata kerja pasif yang digunakan *MC* untuk mengekspresikan kesopanan ketika berkomunikasi dengan orang yang dituju. Kata kerja pasif tidak hanya membuat kalimat menjadi lebih sopan namun juga membuat apa yang disampaikan oleh pembicara lebih harmonis dan menghormati pendengar.

BAB III
METODOLOGI PENELITIAN



Gambar 3.1 Diagram alur penelitian

3.1 Dataset

Penelitian ini menggunakan dataset yang telah digunakan pada penelitian sebelumnya dengan judul "*Semi-Supervised Low-Resource Style Transfer of Indonesian Informal to Formal Language with Iterative Forward-Translation"* (Wibowo et al., 2020). Dataset berisikan total 5000 teks (2500 teks tidak sopan dan 2500 teks sopan). Dataset tersebut memenuhi kriteria batasan masalah pada penelitian ini yaitu menggunakan strategi kesopanan dalam komunikasi dengan cara mempertahankan bentuk formalitas pada kalimat yang disampaikan (Kardana et al., 2018). Kumpulan teks yang terdapat pada dataset merupakan sampel berjumlah 2.500 yang diambil dari 52.000 *tweet* yang telah di-*scrapping* dari sosial media *twiiter* pada penelitian sebelumnya. Dataset dibagi menjadi *training, validation,* dan *test set* yang masing-masing berjumlah 1922, 214, dan 316 teks yang disimpan dalam bentuk file *txt*. Karakteristik teks tidak sopan pada dataset dapat dijelaskan sebagai berikut:

1. **Menggunakan kata sehari-hari atau mempersingkat kata.**
   Contoh: gw knp tdk bs.
   Diubah menjadi: Saya kenapa tidak bisa?
2. **Pengunaan afiks dan sufiks yang tidak sesuai.**
   Contoh: saya mesenin taxi.
   Diubah menjadi: saya memesan taxi.

3. **Urutan kata yang kurang tepat.**

   Contoh: Admin, bisa seperti itu kenapa?

   Diubah menjadi: Admin, kenapa bisa seperti itu?

Setiap kalimat yang terdapat pada dataset sudah di-*preproccess* dengan menggantikan angka menjadi *xxxnumberxxx*, akun yang ditag pada *tweet* menjadi *xxxuserxxx*, dan tanggal menjadi *xxxdatexxx*. Selain itu, kata yang mengandung huruf sama yang berurutan lebih dari dua kali seperti `makannn` dikurangi menjadi hanya dua huruf saja seperti `makann`. Contoh kalimat tidak sopan dan sopan yang terdapat pada Tabel 3.1.

Tabel 3.1 Contoh pasangan kalimat tidak sopan dan sopan pada dataset.

Kalimat Tidak Sopan	Kalimat Sopan
alhamdulillah stlh libur xxxnumberxxx hari onbid lgsg dikasih orderan , food lg . thanks xxxuserxxx cc	alhamdulillah setelah libur xxxnumberxxx hari onbid langsung diberi order , makanan lagi . terima kasih xxxuserxxx cc .
selamat sore min . saya mau pesan tiket ka via web , tetapi selalu tertulis " terjadi kesalahan pada sistem " mohon solusinya . terima kasih	selamat sore admin . saya mau pesan tiket ka via web tetapi selalu tertulis , " terjadi kesalahan pada sistem " mohon solusinya . terima kasih .
iya kak terimakasih . tapi tadi sudah datang ke galeri indosatnya langsung . sudah normal lagi	iya kak terima kasih . tetapi tadi sudah datang ke galeri indosatnya langsung . sudah normal kembali .
malam min xxxuserxxx xxxuserxxx situs kalian error ya . mau order dari siang tadi ga bisa2 .	selamat malam admin xxxuserxxx xxxuserxxx , apakah situs kalian error ? mau order dari siang tadi tidak bisa-bisa .
min pembelian token pln apa ada kendala , ini blm masuk udah xxxnumberxxx jam lebih ?	admin , pembelian token pln apa ada kendala ? ini belum masuk sudah xxxnumberxxx jam lebih .

3.2 Data Preparation

Sebelum melakukan pelatihan model *tagger dan generator* dibutuhkan persiapan dan pemrosesan data agar dapat dilatih dengan baik oleh model *tagger*. Persiapan data yang pertama kali yang dilakukan adalah menggabungkan data *train, val, dan test* pada dataset lalu memberikan label menjadi P_9 untuk teks berjenis sopan dan P_0 untuk teks netral atau tidak sopan menggunakan bahasa pemrograman *Python*. Label gaya teks pada data (P_9 dan P_0) dipilih karena digunakan pada penelitian sebelumnya dan dianggap lebih singkat dan simpel untuk mempermudah proses selanjutnya yang melibatkan penggunaan label. Hasil akhir yang dihasilkan oleh kode *Data Preparation* adalah file berjenis *TSV (Tab Separated Values)* berisikan kumpulan teks dengan label gaya teks (P_0 dan P_9) dan kelas pembagian data pada teks tersebut (*training, dev, atau test*). Selanjutnya, memberikan *tag token* berdasarkan estimasi setiap kata atau frasa pada masing masing gaya teks menggunakan rasio rerata *tf-idf* pada

persamaan (3.1) setelah itu dihitung probabilitas kemunculan *n-gram*-nya menggunakan persamaan (3.2). Selanjutnya, hasil probabilitas yang didapatkan tersebut digunakan untuk menghitung peringkat persentil pada masing-masing *n-gram* di setiap *gaya teks* dengan jangkauan 0% (terendah) sampai 100% (tertinggi). Proses terakhir yang harus dilakukan adalah menetapkan batas atau *threshold* untuk menentukan *n-gram* yang akan dijadikan *token tag* pada gaya teks awal dan *n-gram* yang akan dijadikan sebagai pengisi *token tag* pada gaya teks target. Untuk mempermudah pehamaman diberikan contoh data berikut ini:

1. Teks gaya tidak sopan: ‘min, aq pengen beli pulsa’
2. *Threshold n-gram:* 80%
3. peringkat persentil *n-gram* pada teks terhadap korpus gaya teks tidak sopan:
    a. `min`: 85%
    b. `aq`: 90%
    c. `pengen`: 70%
    d. `beli`:50%
    e. `pulsa`: 20%

Berdasarkan peringkat persentil tersebut, *n-gram* pada teks yang memenuhi atau melebihi *thresold* akan dijadikan sebagai *token tag*. Sehingga, hasil akhir teks: *`[tag], [tag]* pengen beli pulsa`.

$$\eta_1^2(w) = \frac{\frac{1}{m}\sum_{i=1}^{m} tf - i\,df\left(w, x_i^{(2)}\right)}{\frac{1}{n}\sum_{j=1}^{n} tf - idf\left(w, x_j^{(1)}\right)} \quad (3.1)$$

$$p_1^2(w) = \frac{\eta_1^2(w)^\gamma}{\sum_{w'} \eta_1^2(w')^\gamma} \quad (3.2)$$

Diasumsikan diberikan pasangan korpus X_1 dan X_2 dengan gaya teks masing-masing S_1 dan S_2 dan w merupakan *sampling n-grams* yang diambil dari kedua korpus, maka $p_1^2(w)$ menghitung distribusi probabilitas berdasarkan kemunculan kata pada kedua korpus. Secara intuitif, $p_1^2(w)$ merupakan persamaan yang proposional terhadap probabilitas kemunculan suatu *n-gram* pada kedua korpus namun memiliki rerata *tf-idf* yang lebih tinggi pada X_2 dibandingkan X_1. Sedangkan, $\eta_1^2(w)$merupakan rasio dari rerata *tf-idf* pada suatu *n-gram* terhadap korpus X_1 dan X_2. *N-gram* dengan nilai $\eta_1^2(w)$ yang tinggi memiliki rerata *tf-idf* yang tinggi pada korpus X_2 terhadap X_1, yang berarti kata tersebut memiliki karakteristik gaya teks S_2. Setelah semua *n-gram* pada kedua korpus mendapatkan nilai $p_1^2(w)$, didapatkanlah Γ_2

(persamaan (3.3)), yaitu kumpulan *n-gram* pada korpus X_2 dengan nilai $p_1^2(w)$ diatas *thresold* (k) yang telah ditentukan (Madaan et al., 2020).

$$\Gamma_2 = \{w : p_1^2(w)) \geq k\} \quad (3.3)$$

3.3 Training Tagger dan Generator

Dalam melakukan *training,* model *tagger* menerima *input* teks yang berasal dari data *training* sedangkan model *generator* menerima *input* dari *output* pada model *tagger*. Model *tagger* memiliki dua varian model yang setiap variannnya bergantung pada tujuan yang ingin dicapai. Varian pertama ***replace-tagger*** merupakan model yang mengidentifikasi attribut gaya teks pada *input* $\alpha\left(x_1^{(1)}\right)$ lalu menggantikannya dengan *TAG token;* varian ini tepat untuk model yang mempelajari atribut dari kedua gaya teks (*source* dan *target*), seperti misalnya *sentiment transfer* yaitu membutuhkan model untuk mempelajari atribut pada sentimen positif dan negatif. Sedangkan, varian kedua ***add-tagger*** merupakan model yang mengidentifikasi posisi yang tepat untuk menambahkan *token tag* tanpa menghapus kata atau frasa yang ada; varian ini yang dipakai pada penelitian sebelumnya (Madaan et al., 2020). Sebab untuk mengubah kalimat netral menjadi kalimat yang sopan, model hanya perlu mendapatkan informasi tentang atribut dari gaya teks sopan. Secara umum, kedua varian model tersebut memiliki tujuan yang sama yaitu memberikan *tag token* pada teks lalu nantinya akan menjadi *input* untuk model *generator*.

$$L_r(\theta_t) = -\Sigma_{i=1}^{|x_1|} log p_{\theta_t}\left(z(x_i)|x_i^{(1)}; \theta_t\right) \quad (3.4)$$

$$L_a(\theta_t) = -\sum_{i=1}^{|x^1|} log p_\theta t\left(z(x_i)|x_i^{(2)} \backslash a(x_i^{(2)})\right) \quad (3.5)$$

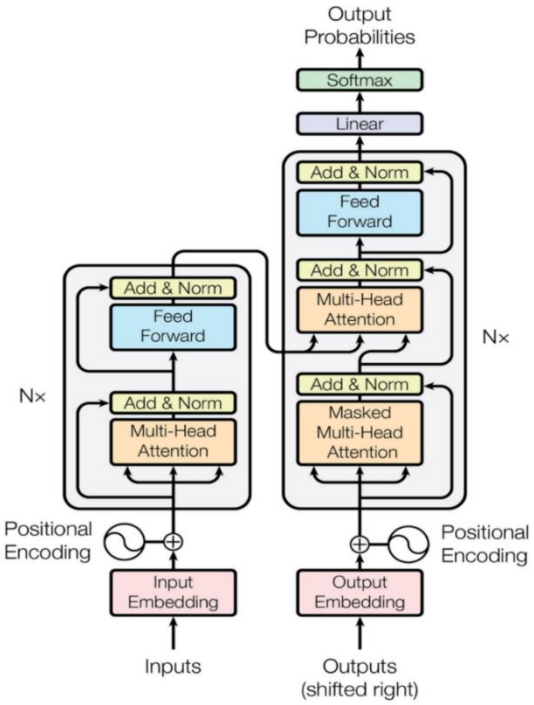
Persamaan (3.4) adalah *loss function* yang digunakan untuk melatih model varian *add-replace* dan persamaan (3.5) untuk model varian *add-tagger.* Pada penelitian ini, model *tagger* membutuhkan informasi atribut pada teks tidak sopan. Oleh sebab itu, model varian *add-replace* yang akan dipakai untuk melatih model *tagger* dan menggunakan persamaan (3.4) sebagai *loss function.*

$$L(\theta_g) = -\Sigma_{i=1}^{|x_v|} log p_{\theta g}\left(x_i^{(v)} \middle| Z(x_i); \theta g\right) \quad (3.6)$$

Sedangkan persamaan (3.6) merupakan *loss function* yang akan dipakai untuk melatih model *generator*. Pada pelatihan model *generator* digunakan juga teknik *data augmentation* berupa:

1. *random word shuffle*: menukar urutan dua kata pada teks dengan probabilitas yang telah ditentukan.
2. *word drops replacement*: menghapus sebuah kata pada teks dengan probabilitas yang telah ditentukan.

Model *tagger* dan *generator* menggunakan 4 lapis model arsitektur *transformer*, di setiap model *transformer* tersebut memiliki 4 *attention heads* dengan 512 *dimensional embedding* dan *hidden state size.* Ditambahkan juga *Dropout* dengan nilai p = 0.3 pada setiap lapisan *transformer*.



Gambar 3.2 Arsitektur lapisan *Transformer*

3.4 Evaluasi

Untuk melakukan evaluasi terhadap performa model, penelitian ini terbatas pada *content preservation* dan *fluency* dari teks yang dihasilkan. Metrik evaluasi yang digunakan untuk

mengukur *content preservation* pada penelitian ini adalah **BLEU** (Papineni et al., 2002b) dan **METEOR** (Lavie & Agarwal, 2007)**.**

BLEU merupakan metrik yang menghitung jumlah kesamaan *n-gram* pada teks yang dihasilkan model dengan teks pada target. Untuk mendapatkan hasil akhir **BLEU**, hal pertama yang dilakukan adalah menghitung rata-rata *geometric modified precision score* pada korpus yang dilakukan pengujian ($\boldsymbol{p_n}$) menggunakan *n-grams* dengan batas N dan bobot (w_n) yang jika dikumulatifkan bernilai 1. Selanjutnya dikalikan oleh *brevity penalty* (BP) atau penalti akibat dari teks yang dihasilkan oleh model lebih pendek dari teks target. Seperti yang dapat dilihat pada persamaan (3.7).

$$BLEU = BP.\exp(\Sigma_{n=1}^{N} w_n log p_n) \quad (3.7)$$

Pada metrik kedua yaitu **METEOR,** metrik tersebut dianggap lebih baik daripada **BLEU** saat menggunakan parameter penilaian manusia dikarenakan metrik tersebut juga mengukur kesamaan semantik tiap *n-gram* pada teks yang dihasilkan oleh model dan teks target. **METEOR** merupakan modifikasi dari perhitungan *precision* dan *recall* dengan *weighted F-Score* berdasarkan pemetaan *unigram* pada teks yang dihasilkan model dengan teks target.

$$F = \frac{PR}{\alpha P + (1 - a)R} \quad (3.8)$$

Untuk menghitung *weighted F-Score,* yang pertama kali dilakukan adalah menghitung jumlah pemetaan (m) *unigram* pada kedua teks (teks yang dihasilkan oleh model dan teks target). Setelah itu dilakukan penghitungan *precision: m/c* dan *recall: m/r*, yang mana c adalah panjang teks yang dihasilkan model dan r adalah panjang teks pada target. Hasil akhir *weighted F-Score* dapat dihitung menggunakan persamaan (3.8).

Berbeda dengan **BLEU, METEOR** menghitung *penalty* berdasarkan urutan *n-gram* pada teks yang dihasilkan model tidak sesuai dengan teks target. Hasil akhir dari skor **METEOR** didapatkan melalui (1 - *Penalty*) x *F* dengan jangkauan *range* skor dari 0 sampai 1. Dapat disimpulkan bahwa semakin tinggi skor **METEOR** maka semakin banyak kesamaan *n-gram* pada teks yang dihasilkan oleh model dengan teks pada target (berdasarkan kecocokan *n-gram* dan urutannya)

BAB IV
HASIL DAN PEMBAHASAN

4.1 Kode *Data Preparation*

Hal pertama yang dilakukan dalam *data preparation* adalah melabeli setiap kalimat pada dataset dengan label P_0 untuk kalimat tidak sopan dan P_9 untuk kalimat sopan. *Labelling* dilakukan dengan membuat tiga *list*:

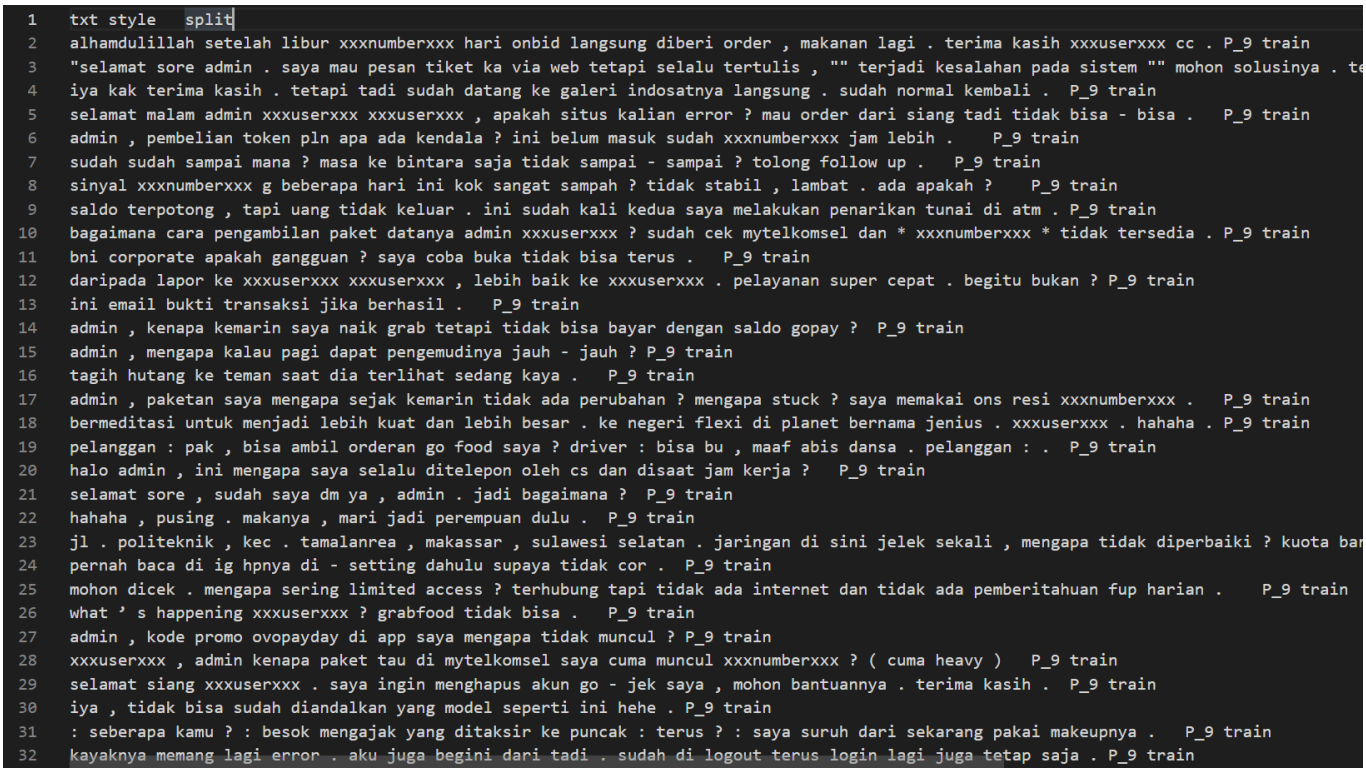
1. *allData*: berisikan seluruh teks dari ketiga berkas dataset (*training*, *validation*, *dan test*).
2. *labels:* berisikan label jenis gaya teks (P_0 untuk teks tidak sopan dan P_9 untuk teks sopan).
3. *datasplit:* berisikan label asal dataset dari teks (*training*, *validation*, atau *test*).

Ketiga *list* tersebut dibuat secara paralel menggunakan iterasi *for loop* agar *index-nya* sesuai dengan urutan teks yang terdapat masing-masing berkas dataset. Selanjutnya adalah, melakukan pembuatan *dataframe* dengan bantuan *library pandas* yang nantinnya akan memudahkan pembuatan *file* berbentuk *TSV (tab separated value)* sebagai *output*-nya. Kode yang digunakan untuk melakukan pelabelan data dapat dilihat pada Gambar 4.1 dan hasil *file TSV* pada Gambar 4.2.

```
import pandas as pd
labels = []
dataSplit = []

with open('kalimat_sopan_training.txt', 'r', encoding='utf-8') as f :
    lst = f.readlines()
    polites_train =[]
    for x in lst:
        polites_train.append(x.replace("\n", ""))
    f.close()
labels = labels + ['P_9'] * len(polites_train)
dataSplit = dataSplit + ['train'] * len(polites_train)

with open('kalimat_tidak_sopan_training.txt', 'r', encoding='utf-8') as f :
    lst = f.readlines()
    impolites_train =[]
    for x in lst:
        impolites_train.append(x.replace("\n", ""))
    f.close()
labels = labels + ['P_0'] * len(impolites_train)
dataSplit = dataSplit + ['train'] * len(impolites_train)

with open('kalimat_sopan_val.txt', 'r', encoding='utf-8') as f :
    lst = f.readlines()
    polites_dev =[]
    for x in lst:
        polites_dev.append(x.replace("\n", ""))
```

```
    f.close()
labels = labels + ['P_9'] * len(polites_dev)
dataSplit = dataSplit + ['val'] * len(polites_dev)
print(dataSplit)

with open('kalimat_tidak_sopan_val.txt', 'r', encoding='utf-8') as f :
    lst = f.readlines()
    impolites_dev =[]
    for x in lst:
        impolites_dev.append(x.replace("\n", ""))
    f.close()
labels = labels + ['P_0'] * len(impolites_dev)
dataSplit = dataSplit + ['val'] * len(impolites_dev)


with open('kalimat_sopan_test.txt', 'r', encoding='utf-8') as f :
    lst = f.readlines()
    polites_test =[]
    for x in lst:
        polites_test.append(x.replace("\n", ""))
    f.close()
labels = labels + ['P_9'] * len(polites_test)
dataSplit = dataSplit + ['test'] * len(polites_test)

with open('kalimat_tidak_sopan_test.txt', 'r', encoding='utf-8') as f :
    lst = f.readlines()
    impolites_test =[]
    for x in lst:
        impolites_test.append(x.replace("\n", ""))
    f.close()

labels = labels + ['P_0'] * len(impolites_test)
dataSplit = dataSplit + ['test'] * len(impolites_test)


allData = pd.DataFrame()
allData['txt']= polites_train +impolites_train + polites_dev + impolites_dev
+ polites_test + impolites_test
allData['style'] = labels
allData['split'] =  dataSplit
allData.to_csv('data.tsv',sep = '\t', index=False)
```

Gambar 4.1 Kode pelabelan pada dataset



```
txt style   split
alhamdulillah setelah libur xxxnumberxxx hari onbid langsung diberi order , makanan lagi . terima kasih xxxuserxxx cc . P_9 train
"selamat sore admin . saya mau pesan tiket ka via web tetapi selalu tertulis , "" terjadi kesalahan pada sistem "" mohon solusinya . te
iya kak terima kasih . tetapi tadi sudah datang ke galeri indosatnya langsung . sudah normal kembali .  P_9 train
selamat malam admin xxxuserxxx xxxuserxxx , apakah situs kalian error ? mau order dari siang tadi tidak bisa - bisa .   P_9 train
admin , pembelian token pln apa ada kendala ? ini belum masuk sudah xxxnumberxxx jam lebih .    P_9 train
sudah sudah sampai mana ? masa ke bintara saja tidak sampai - sampai ? tolong follow up .   P_9 train
sinyal xxxnumberxxx g beberapa hari ini kok sangat sampah ? tidak stabil , lambat . ada apakah ?    P_9 train
saldo terpotong , tapi uang tidak keluar . ini sudah kali kedua saya melakukan penarikan tunai di atm . P_9 train
bagaimana cara pengambilan paket datanya admin xxxuserxxx ? sudah cek mytelkomsel dan * xxxnumberxxx * tidak tersedia . P_9 train
bni corporate apakah gangguan ? saya coba buka tidak bisa terus .   P_9 train
daripada lapor ke xxxuserxxx xxxuserxxx , lebih baik ke xxxuserxxx . pelayanan super cepat . begitu bukan ? P_9 train
ini email bukti transaksi jika berhasil .   P_9 train
admin , kenapa kemarin saya naik grab tetapi tidak bisa bayar dengan saldo gopay ?  P_9 train
admin , mengapa kalau pagi dapat pengemudinya jauh - jauh ? P_9 train
tagih hutang ke teman saat dia terlihat sedang kaya .   P_9 train
admin , paketan saya mengapa sejak kemarin tidak ada perubahan ? mengapa stuck ? saya memakai ons resi xxxnumberxxx .   P_9 train
bermeditasi untuk menjadi lebih kuat dan lebih besar . ke negeri flexi di planet bernama jenius . xxxuserxxx . hahaha . P_9 train
pelanggan : pak , bisa ambil orderan go food saya ? driver : bisa bu , maaf abis dansa . pelanggan : .  P_9 train
halo admin , ini mengapa saya selalu ditelepon oleh cs dan disaat jam kerja ?   P_9 train
selamat sore , sudah saya dm ya , admin . jadi bagaimana ?  P_9 train
hahaha , pusing . makanya , mari jadi perempuan dulu .  P_9 train
jl . politeknik , kec . tamalanrea , makassar , sulawesi selatan . jaringan di sini jelek sekali , mengapa tidak diperbaiki ? kuota ban
pernah baca di ig hpnya di - setting dahulu supaya tidak cor .  P_9 train
mohon dicek . mengapa sering limited access ? terhubung tapi tidak ada internet dan tidak ada pemberitahuan fup harian .    P_9 train
what ' s happening xxxuserxxx ? grabfood tidak bisa .   P_9 train
admin , kode promo ovopayday di app saya mengapa tidak muncul ? P_9 train
xxxuserxxx , admin kenapa paket tau di mytelkomsel saya cuma muncul xxxnumberxxx ? ( cuma heavy )   P_9 train
selamat siang xxxuserxxx . saya ingin menghapus akun go - jek saya , mohon bantuannya . terima kasih .  P_9 train
iya , tidak bisa sudah diandalkan yang model seperti ini hehe . P_9 train
: seberapa kamu ? : besok mengajak yang ditaksir ke puncak : terus ? : saya suruh dari sekarang pakai makeupnya .   P_9 train
kayaknya memang lagi error . aku juga begini dari tadi . sudah di logout terus login lagi juga tetap saja . P_9 train
```

Gambar 4.2 Hasil dari kode pelabelan data berbentuk *TSV*

Setelah mendapatkan hasil dari pelabelan data berbentuk *TSV*, *file* tersebut akan menjadi *input* untuk tahap pemilihan *n-gram* atau kata yang akan dijadikkan *token tag.* Untuk mendapatkan *token tag* yang akan dipakai model *tagger*, hal pertama yang dilakukan adalah menghitung *TF-IDF* setiap *n-gram* pada kedua gaya teks. Hal tersebut diimplementasikan dalam bentuk *Class TFIDFStatsGenerator* seperti kode pada Gambar 4.3 yang pada saat inisiasinya membutuhkan parameter yang harus dipenuhi seperti:

1. *n-gram_range*: jangkauan minimal dan maksimal jumlah *n-gram* yang akan digunakan dalam bentuk *tuple* (penelitian menggunakan jangkauan *unigram* sampai *bigram* sehingga nilainya menjadi (1,2)).
2. *data_id:* label jenis gaya pada teks yang digunakan dalam bentuk *string* (P_0 untuk teks tidak sopan dan P_9 untuk teks sopan).
3. *data*: data yang akan digunakan dalam bentuk *pandas dataframe.*

Setelah semua variabel tersebut dipenuhi, selanjutnya dilakukan penghitungan *TF-IDF* menggunakan fungsi *generate* dengan bantuan fungsi *TfidfVectorizer* dari *library scikit learn.*

Hasil akhir dari *class* ini adalah objek dengan variabel yang nantinya dibutuhkan untuk proses selanjutnya seperti:

1. *tfidf_avg*: berisikan jumlah rerata *tfid* pada data.
2. *word_to_idf:* berisikan *n-gram* dan nilai *tfidf*-nya dalam bentuk struktur data *dictionary.*
3. *count:* berisikan *n-gram* dan jumlahnya pada data dalam bentuk struktur data *dictionary.*

```
class TFIDFStats Generator:

    def __init__(self, data, data_id, ngram_range):
        super().__init__()
        self.ngram_range = ngram_range
        self.data_id = data_id
        self.data = data
        self.generate()

    def get_word_counts(self):
        """Generates the counts for various n-grams for the given corpus

        Returns:
            a dictionary from phrase to word count
        """
        cv = CountVectorizer(ngram_range=self.ngram_range)
        cv_fit = cv.fit_transform(self.data)
        feature_names = cv.get_feature_names()
        X = np.asarray(cv_fit.sum(axis=0)) # sum counts across sentences
        word_to_id = {feature_names[i]: i for i in
range(len(cv.get_feature_names()))}
        word_count = {}
        for w in word_to_id:
            word_count[w] = X[0, word_to_id[w]]
        return word_count

    def generate(self):
        """Generates various TFIDF related stats
        for the given data and wraps them in a namedtuple

        Returns:
            [type] -- [description]
        """
        logging.info("Running TfidfVectorizer")
        vectorizer = TfidfVectorizer(ngram_range=self.ngram_range)
        X = vectorizer.fit_transform(self.data)
        feature_names = vectorizer.get_feature_names()
        id_to_word = {i: feature_names[i] for i in
range(len(vectorizer.get_feature_names()))}
        word_to_id = {v: k for k, v in id_to_word.items()}
        X = np.asarray(X.mean(axis=0)).squeeze(0) # / num_docs

        idf = vectorizer.idf_
        counts = self.get_word_counts()
```

```
        word_to_idf = dict(zip(feature_names, idf))

        self.id_to_word = id_to_word
        self.word_to_id = word_to_id
        self.tfidf_avg = X
        self.word_to_idf = word_to_idf
        self.counts = counts
```

Gambar 4.3 Kode untuk *Class TFIDFStatsGenerator*

Tahap selanjutnya setelah mendapatkan objek dari *Class TFIDFStatsGenerator* adalah melanjutkan penghitungan rasio rerata relatif *tf-idf n-gram* dari suatu jenis gaya teks terhadap gaya teks lainnya seperti yang telah dijelaskan pada persamaan (3.1). Untuk melakukan penghitungan tersebut dibuatlah sebuah *class* bernama *RelativeTagsGenerator* seperti yang dapat dilihat pada Gambar 4.4. Untuk dapat memanggil *class* tersebut dibutuhkan parameter yang harus dipenuhi seperti:

1. *main_class_stats* dan *relative_class_stats:* objek *class TFIDFStatsGenerator* dari gaya teks utama dan gaya teks relatifnya. Misal: untuk menghitung rasio rerata relatif *tf-idf n-gram* dari gaya teks sopan relatif terhadap gaya teks tidak sopan, maka *main_class_stats* merupakan hasil dari *TFIDFStatsGenerator* dari gaya teks P_9 dan *relative_class* dari gaya teks P_0.
2. *min_freq:* jumlah minimum frekuensi *n-gram* yang digunakan untuk penghitungan rasio rerata *tf-idf* (*default* = 2).
3. *thresh:* skor relatif *tf-idf* nantinya akan diubah menjadi peringkat persentil dan parameter ini digunakan sebagai batas minimal peringkat persentil *n-gram* yang akan dipertahankan menjadi *tag-token.*
4. *Ignore_from_tags:* kumpulan *n-gram* yang perlu diabaikan atau tidak akan dijadikan *token tag* dalam berbentuk data struktur *list.*

Setelah semua parameter telah dipenuhi, *class* akan memanggil fungsi *generate_tfidf_report* dan *generate_relative_tags* yang digunakan untuk membuat *pandas dataframe* dengan nama variabel *report* yang berisi kolom:

1. *word:* berisikan *n-gram* yang diambil dari variabel *main_class_stats.*
2. *freq:* berisikan frekuensi *n-gram* pada kolom *word.*
3. {gaya teks utama}*_mean_tfidf:* berisikan angka rerata *tfidf n*-gram di kolom *word* pada korpus gaya teks utama.

4. {gaya teks relatif}*_mean_tfidf:* berisikan angka rerata *tfidf n*-gram di kolom *word* pada korpus gaya teks relatif.
5. {gaya teks utama}*_idf:* berisikan nilai *idf n*-gram di kolom *word* pada korpus gaya teks utama.
6. {gaya teks relatif}*_idf:* berisikan nilai *idf n*-gram di kolom *word* pada korpus gaya teks relatif.
7. {gaya teks utama}*_over_*{gaya teks relatif}: berisikan nilai rasio rerata relatif *tf-idf n-gram* dari gaya teks utama terhadap gaya teks relatif.
8. {gaya teks relatif}*_over_*{gaya teks utama}: berisikan nilai rasio rerata relatif *tf-idf n-gram* di kolom *word* dari gaya teks relatif terhadap gaya teks utama.
9. {gaya teks utama}*_tag*: berisikan nilai probabilitas kemunculan *n-gram* pada kolom *word* di korpus gaya teks utama dan relatif.
10. *rank:* berisikan peringkat persentil *n-gram* di kolom *word* berdasarkan nilai {gaya teks utama}*_tag*.

Hasil akhir yang didapatkan setelah pemanggilan *class* ini adalah kumpulan *n-gram* yang peringkat persentilnya telah melewati atau sama dengan batas yang telah ditentukan melalui variabel *thresh*. Untuk selanjutnya, hasil tersebut digunakan untuk menyusun data yang digunakan untuk melatih model *tagger* dan model *generator*.

```
class RelativeTags Generator:

    def __init__(self, main_class_stats, relative_class_stats,
                 min_freq: int = 2, thresh: float = 0.70,
                 ignore_from_tags = None):

        super().init__()
        self.main_class_stats = main_class_stats
        self.relative_class_stats = relative_class_stats
        self.min_freq = min_freq
        self.c1_tag = main_class_stats.data_id
        self.c2_tag = relative_class_stats.data_id
        self.thresh = thresh
        self.ignore_from_tags = ignore_from_tags

        self.generate_tfidf_report()
        self.generate_relative_tags()




    def generate_tfidf_report(self):

        report = []
        for word in self.main_class_stats.word_to_id.keys():
            if  self.main_class_stats.counts[word]  >=  self.min_freq  and  word  in
self.relative_class_stats.word_to_id:
                    res = {}
                    res["word"] = word
```

```
                    res["freq"] = self.main_class_stats.counts[word]
                    res[f"{self.c1_tag}_mean_tfidf"]                                    =
self.main_class_stats.tfidf_avg[self.main_class_stats.word_to_id[word]]
                    res[f"{self.c2_tag}_mean_tfidf"]                                    =
self.relative_class_stats.tfidf_avg[self.relative_class_stats.word_to_id[word]]
                    res[f"{self.c1_tag}_idf"]                                           =
self.main_class_stats.word_to_idf[word]
                    res[f"{self.c2_tag}_idf"]                                           =
self.relative_class_stats.word_to_idf[word]
                    report.append(res)
        self.report = pd.DataFrame(report)




    def generate_relative_tags(self):
        """Returns a dictionary of phrases that are important in class1 relative to
        class2
        """
        c1_over_c2 = f"{self.c1_tag}_over_{self.c2_tag}"
        c2_over_c1 = f"{self.c2_tag}_over_{self.c1_tag}"
        #    tfidf_report["np_over_p"]    =    (tfidf_report["np_mean_tfidf"]    /
len(data_p_0)) / (tfidf_report["p_mean_tfidf"] /  len(data_p_9))
        self.report[c1_over_c2]    =    self.report[f"{self.c1_tag}_mean_tfidf"]    /
self.report[f"{self.c2_tag}_mean_tfidf"] #ratio of tf-idf in the two corpora

        self.report[c2_over_c1] = 1 / self.report[c1_over_c2]

        self.report[f"{self.c1_tag}_tag"]     =     (self.report[c1_over_c2]     /
self.report[c1_over_c2].sum()) ** 0.75
        # ^ add support for the small values

        self.report[f"{self.c1_tag}_tag"]   =   self.report[f"{self.c1_tag}_tag"]   /
self.report[f"{self.c1_tag}_tag"].sum()
        # ^ make a probability

        self.report.sort_values(by=f"{self.c1_tag}_tag",           ascending=False,
inplace=True)
        self.report['rank'] = self.report[f"{self.c1_tag}_tag"].rank(pct=True)
        # ^ assign percentile
        self.report.to_csv('report.csv')


        important_phrases = self.report[self.report["rank"] >= self.thresh]
        # ^ only take phrases that clear the threshold (default: 0.9)

        important_phrases["score"] = (important_phrases["rank"] - self.thresh) / (1
- self.thresh)
        # ^ make a distribution again

        tags= {}
        for i, r in important_phrases.iterrows():
            tags[r["word"]] = r["score"]

        self.tags = tags

        if self.ignore_from_tags is not None:
            logging.info("Ignoring tags")
            self.tags = self.filter_tags_with_ignored_entities()




    def filter_tags_with_ignored_entities(self):
        res = {}
        for k, v in self.tags.items():
```

```
            if not any(k_part in self.ignore_from_tags for k_part in k.split()):
                res[k] = v
        return res
```

Gambar 4.4 Kode untuk *class RelativeTagsGenerator*

Kode pada Gambar 4.5 merupakan keseluruhan proses dari *data preparation* yang membutuhkan *class RelativeTagsGenerator* dan *TFIDFStatsGenerator* untuk melakukan tugasnya. Kode tersebut memiliki dua tahapan dalam melakukan tugasnya, yaitu:

1. Menghasilkan *file JSON* seperti pada Gambar 4.6 dan 4.7 berisikan *n-gram* yang telah memenuhi standar *threshold* peringkat persentil yang telah ditentukan di *class RelativeTagsGenerator*.
2. Men-*generate dataset training*, *validation*, dan *test* untuk pelatihan model *tagger* dan *generator*. Data yang dihasilkan merupakan data paralel: satu data berformat *tagged* atau yang berisikan kalimat yang *n-gram*-nya telah digantikan dengan *token tag* dan data yang berisikan kalimat utuh secara keseluruhan.

```
from docopt import docopt
import json
import pandas as pd
import subprocess
import logging

from style_tags import TFIDFStats*Generator*, RelativeTags*Generator*, TrainDataGen


def tag_style_markers(data_pth: str, outpath: str, style_0_label: str, style_1_label:
str, tgt_lang="tagged", thresh=0, ngram_range=(1, 2),
                      ignore_from_tags=None,               style_label_col="label",
drop_duplicates=False,
                      gen_tags=True):

    data = pd.read_csv(data_pth, sep="\t")
    if drop_duplicates:
        data = data.drop_duplicates(subset="txt")

    logging.info("Reading the data")
    data_style_0 = data[data[style_label_col] == style_0_label]
    data_style_1 = data[data[style_label_col] == style_1_label]

    if gen_tags:
        logging.info("Getting TF-IDF stats for both the corpora")
        logging.info(f"#Records {style_0_label} = {len(data_style_0)}")
        logging.info(f"#Records {style_1_label} = {len(data_style_1)}")

        tags_style_0,                          tags_style_1                          =
generate_tags(df_txt_class_1=data_style_0[data_style_0["split"] != "test"]["txt"],

df_txt_class_2=data_style_1[data_style_1["split"]
                                                                                !=
"test"]["txt"],
                                                tag_class_1=style_0_label,
                                                tag_class_2=style_1_label,
```

```
ignore_from_tags=ignore_from_tags,
                                                    thresh=thresh,
                                                    ngram_range=ngram_range)

        with open(f"{outpath}/{style_0_label}_tags.json", "w") as f:
            json.dump(tags_style_0, f)

        with open(f"{outpath}/{style_1_label}_tags.json", "w") as f:
            json.dump(tags_style_1, f)

    else:
        with open(f"{outpath}/{style_0_label}_tags.json", "r") as f:
            tags_style_0 = json.load(f)
        with open(f"{outpath}/{style_1_label}_tags.json", "r") as f:
            tags_style_1 = json.load(f)

    # Step 3
    logging.info("Generating the tagged data")
    TrainDataGen(data=data_style_0, outpath=outpath, tags=tags_style_0,
                 tag_token=style_0_label, tgt_lang=tgt_lang).generate()
    TrainDataGen(data=data_style_1, outpath=outpath, tags=tags_style_1,
                 tag_token=style_1_label, tgt_lang=tgt_lang).generate()


def generate_tags(df_txt_class_1,
                  df_txt_class_2,
                  tag_class_1,
                  tag_class_2,
                  thresh,
                  ngram_range,
                  ignore_from_tags=None,
                  ):
    stats_class_1 = TFIDFStats*Generator*(
        df_txt_class_1, tag_class_1, ngram_range=ngram_range)
    stats_class_2 = TFIDFStats*Generator*(
        df_txt_class_2, tag_class_2, ngram_range=ngram_range)

    class_1_tags = RelativeTags*Generator*(main_class_stats=stats_class_1,
                                           relative_class_stats=stats_class_2,
                                           ignore_from_tags=ignore_from_tags,
                                           thresh=thresh).tags

    class_2_tags = RelativeTags*Generator*(main_class_stats=stats_class_2,
                                           relative_class_stats=stats_class_1,
                                           thresh=thresh).tags
    return class_1_tags, class_2_tags


def prepare_parallel_data_*tagger*(outdir, style_0_label, style_1_label, is_unimodal):
    subprocess.check_call(f"scripts/prep_*tagger*.sh    {outdir}    {outdir}    tagged
{int(is_unimodal)} {style_0_label} {style_1_label}",
                          shell=True)


def    prepare_parallel_data_*generator*(outdir,    style_0_label,    style_1_label,
is_unimodal):
    #   "${MASKED_OP_DIR}"   "${MASKED_OP_DIR}"   "$prefix"masked   "$prefix"unmasked
"$isunimodal" "$posmask" "$negmask"
    subprocess.check_call(f"scripts/prep_*generator*.sh   {outdir}   {outdir}   tagged
generated {int(is_unimodal)} {style_0_label} {style_1_label}",
                          shell=True)


if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO)
```

```
    args = docopt(__doc__)
    is_unimodal = int(args["--is_unimodal"] == "True")

    tag_style_markers(data_pth=args["--data_pth"],
                      outpath=args["--outpath"],
                      style_0_label=args["--style_0_label"],
                      style_1_label=args["--style_1_label"],
                      thresh=float(args["--thresh"]),
                      ngram_range=(int(args["--ngram_range_min"]),
                                   int(args["--ngram_range_max"])),
                      style_label_col=args["--style_label_col"],
                      gen_tags=(args["--gen_tags"] == "True"))



    prepare_parallel_data_tagger(
        args["--outpath"],     args["--style_0_label"],     args["--style_1_label"],
is_unimodal)

    prepare_parallel_data_generator(
        args["--outpath"],     args["--style_0_label"],     args["--style_1_label"],
is_unimodal)
```

Gambar 4.5 Kode lengkap *Data Preparation*

```
{
{
    "mengapa": 1.0,
    "bagaimana ini": 0.9971214738054115,
    "mengapa saya": 0.994242947610823,
    "namun": 0.9913644214162345,
    "tapi tidak": 0.988485895221646,
    "seperti itu": 0.9856073690270586,
    "admin mau": 0.9827288428324701,
    "xxxuserxxx mengapa": 0.9798503166378816,
    "ini bagaimana": 0.9769717904432931,
    "saja tidak": 0.9740932642487046,
    "tidak dibalas": 0.9712147380541161,
    "membuat": 0.9683362118595276,
    "yang lalu": 0.9654576856649391,
    "kalau aku": 0.9625791594703517,
    "tahu": 0.9597006332757632,
    "tolong ditindaklanjuti": 0.9568221070811747,
    "tetapi": 0.9539435808865862,
...
}
```

Gambar 4.6 File *JSON* peringkat persentil *n-gram* pada gaya teks sopan

```
{
    "kalo": 1.0,
    "udah": 0.9967148488830482,
    "ko": 0.9934296977660975,
    "ya min": 0.9901445466491456,
    "ini gimana": 0.9868593955321949,
    "bener": 0.9835742444152431,
    "aja": 0.9802890932982913,
    "gitu": 0.9770039421813406,
    "rb": 0.9720762155059128,
    "xxxnumberxxx rb": 0.9720762155059128,
```

```
    "dipake": 0.9671484888304862,
    "gua": 0.9638633377135344,
    "kepotong": 0.9605781865965837,
    "saya ya": 0.9572930354796319,
    "kok": 0.9540078843626811,
    "emang": 0.9507227332457293,
    "mimin": 0.9474375821287775,
    "abis": 0.9441524310118268,
    "makasih": 0.940867279894875,
    "gk": 0.9375821287779242,
    "dong min": 0.9342969776609724,
    "no": 0.9310118265440206,
    "min mau": 0.9277266754270699,
    "kaya": 0.9244415243101181,
    "tak": 0.9211563731931673,
...
}
```

Gambar 4.7 File *JSON* peringkat persentil *n-gram* pada gaya teks tidak sopan

4.2 Kode *Training* Model

Gambar 4.8 menunjukkan kode yang digunakan untuk melatih model *tagger* dan *generator*. Untuk membedakan jenis model mana yang akan dilatih (*tagger* atau *generator*), parameter yang harus dibedakan adalah di bagian *tgt* atau target dari data yang akan digunakan sebagai pelatihan model. Jika ingin melatih model *generator* maka parameter tersebut diisi dengan nilai *generated*, sedangkan untuk model *tagger* diisi dengan nilai *tagged.* Selain itu terdapat parameter lain yang perlu dipertimbangkan dalam melatih model, seperti:

1. *n-layers* = jumlah lapisan *transformer* yang digunakan untuk membentuk model.
2. *n-heads* = jumlah *attention heads* yang digunakan pada lapisan *transformer.*
3. *embed-dim* = jumlah *dimensional embedding* pada lapisan *transformer.*
4. *Hidden-dim* = jumlah *hidden embedding* yang digunakan untuk membentuk model.
5. *Dropout* = jumlah *frequency rate* yang digunakan untuk *Dropout Layer.*
6. *lr =Learning rate* pada model.
7. *n-epoch* = jumlah *epoch* yang digunakan untuk melatih model.

```
tgt="$1"
dataset="$2"
base_folder="$3"

# Switch to 0 for no bpe
BPE=0
if [ "$BPE" -eq  1 ]; then
    MODEL_PTH=models/$dataset/"bpe"
    echo "Using BPE"
else
    MODEL_PTH=models/$dataset/"nobpe"
    echo "Not using BPE"
```

```
fi

mkdir -p $MODEL_PTH

if [ "$BPE" -eq 1 ]; then
    python src/training.py \
        --src en \
        --tgt "$tgt" \
        --model-file "$MODEL_PTH/en-${tgt}-tagger.pt" \
        --n-layers 4 \
        --n-heads 4 \
        --embed-dim 512 \
        --hidden-dim 512 \
        --dropout 0.3 \
        --bpe \
        --word-dropout 0.1 \
        --lr 1e-3 \
        --n-epochs 15 \
        --tokens-per-batch 8000 \
        --clip-grad 1.1 \
        --base-folder "$base_folder"
else
    python src/training.py \
        --src en \
        --tgt "$tgt" \
        --model-file "$MODEL_PTH/en-${tgt}-tagger.pt" \
        --n-layers 4 \
        --n-heads 4 \
        --embed-dim 512 \
        --hidden-dim 512 \
        --dropout 0.3 \
        --word-dropout 0.1 \
        --lr 1e-3 \
        --n-epochs 5 \
        --tokens-per-batch 8000 \
        --clip-grad 1.1 \
        --base-folder "$base_folder"
fi
```

Gambar 4.8 Kode *bash* yang digunakan untuk menjalakan program *python training* model *tagger* dan *generator.*

4.3 Kode Evaluasi METEOR

Kode pada Gambar 4.9 merupakan implementasi dari metrik METEOR menggunakan *stemmer* berbahasa Indonesia dengan bantuan *library* Sastrawi. *Stemmer* adalah fungsi yang dapat mengubah sebuah kata menjadi bentuk kata dasarnya, contoh: kata ’memakan’ diubah bentuk kata dasarnya menjadi ‘makan’. *Stemmer* tersebut nantinya dapat membantu menghitung kesamaan kata dasar pada teks yang dihasilkan oleh model dan teks target. Kode tersebut memiliki alur sebagai berikut:

1. Fungsi *meteor_score* menerima *input* wajib yaitu teks yang dihasilkan model dalam variabel *hypothesis* dan teks target yang digunakan untuk acuan dalam variabel *reference.*

2. Mencari kesamaan kata pada teks yang terdapat pada variabel *hypothesis* dan *reference* menggunakan fungsi *_enum_align_words*. Kata yang dibandingkan pada kedua variabel tersebut tidak hanya kata utuh yang terdapat pada teks namun juga kata yang telah diubah menggunakan bentuk dasar menggunakan fungsi *stemmer*. Hasil akhir yang didapatkan berbentuk data struktur *list* dengan nama variabel *matches* yang berisikan kata yang sama pada kedua teks tersebut.
3. Menghitung *chunk penalty* atau penalti yang diakibatkan oleh urutan kata yang tidak sesuai pada teks yang dihasilkan model dengan teks target yang disimpan dalam variabel *penalty*.
4. Menghitung *precision*, *recall*, dan *weighted F-Score* yang nantinya didapatkan hasil akhir dari metrik METEOR yaitu : (1 – *chunk penalty*) x *weighted F-Score.*

```
from itertools import chain, product
from typing import Callable, Iterable, List, Tuple

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory


def _generate_enums(
    hypothesis: Iterable[str],
    reference: Iterable[str],
    preprocess: Callable[[str], str] = str.lower,
) -> Tuple[List[Tuple[int, str]], List[Tuple[int, str]]]:

    if isinstance(hypothesis, str):
        raise TypeError(
            f'"hypothesis"   expects   pre-tokenized   hypothesis   (Iterable[str]):
{hypothesis}'
        )

    if isinstance(reference, str):
        raise TypeError(
            f'"reference"   expects   pre-tokenized   reference   (Iterable[str]):
{reference}'
        )

    enum_hypothesis_list = list(enumerate(map(preprocess, hypothesis)))
    enum_reference_list = list(enumerate(map(preprocess, reference)))
    return enum_hypothesis_list, enum_reference_list


def exact_match(
    hypothesis: Iterable[str], reference: Iterable[str]
) -> Tuple[List[Tuple[int, int]], List[Tuple[int, str]], List[Tuple[int, str]]]:

    enum_hypothesis_list,    enum_reference_list    =    _generate_enums(hypothesis,
reference)
    return _match_enums(enum_hypothesis_list, enum_reference_list)



def _match_enums(
    enum_hypothesis_list: List[Tuple[int, str]],
    enum_reference_list: List[Tuple[int, str]],
```

```
) -> Tuple[List[Tuple[int, int]], List[Tuple[int, str]], List[Tuple[int, str]]]:

    word_match = []
    for i in range(len(enum_hypothesis_list))[::-1]:
        for j in range(len(enum_reference_list))[::-1]:
            if enum_hypothesis_list[i][1] == enum_reference_list[j][1]:
                word_match.append(
                    (enum_hypothesis_list[i][0], enum_reference_list[j][0])
                )
                enum_hypothesis_list.pop(i)
                enum_reference_list.pop(j)
                break
    return word_match, enum_hypothesis_list, enum_reference_list


def _enum_stem_match(
    enum_hypothesis_list: List[Tuple[int, str]],
    enum_reference_list: List[Tuple[int, str]],
) -> Tuple[List[Tuple[int, int]], List[Tuple[int, str]], List[Tuple[int, str]]]:

    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    stemmed_enum_hypothesis_list = [
        (word_pair[0],     stemmer.stem(word_pair[1]))     for     word_pair     in
enum_hypothesis_list
    ]

    stemmed_enum_reference_list = [
        (word_pair[0],     stemmer.stem(word_pair[1]))     for     word_pair     in
enum_reference_list
    ]

    return _match_enums(stemmed_enum_hypothesis_list, stemmed_enum_reference_list)


def stem_match(
    hypothesis: Iterable[str],
    reference: Iterable[str],
) -> Tuple[List[Tuple[int, int]], List[Tuple[int, str]], List[Tuple[int, str]]]:

    enum_hypothesis_list,    enum_reference_list    =    _generate_enums(hypothesis,
reference)
    return _enum_stem_match(enum_hypothesis_list, enum_reference_list)


def _enum_align_words(
    enum_hypothesis_list: List[Tuple[int, str]],
    enum_reference_list: List[Tuple[int, str]],
) -> Tuple[List[Tuple[int, int]], List[Tuple[int, str]], List[Tuple[int, str]]]:

    exact_matches, enum_hypothesis_list, enum_reference_list = _match_enums(
        enum_hypothesis_list, enum_reference_list
    )

    stem_matches, enum_hypothesis_list, enum_reference_list = _enum_stem_match(
        enum_hypothesis_list, enum_reference_list
    )


    return (
        sorted(
            exact_matches + stem_matches, key=lambda wordpair: wordpair[0]
        ),
        enum_hypothesis_list,
        enum_reference_list,
    )
```

```
def align_words(
    hypothesis: Iterable[str],
    reference: Iterable[str],
) -> Tuple[List[Tuple[int, int]], List[Tuple[int, str]], List[Tuple[int, str]]]:

    enum_hypothesis_list,    enum_reference_list    =    _generate_enums(hypothesis,
reference)
    return _enum_align_words(
        enum_hypothesis_list, enum_reference_list)



def _count_chunks(matches: List[Tuple[int, int]]) -> int:

    i = 0
    chunks = 1
    while i < len(matches) - 1:
        if (matches[i + 1][0] == matches[i][0] + 1) and (
            matches[i + 1][1] == matches[i][1] + 1
        ):
            i += 1
            continue
        i += 1
        chunks += 1
    return chunks


def single_meteor_score(
    reference: Iterable[str],
    hypothesis: Iterable[str],
    preprocess: Callable[[str], str] = str.lower,
    alpha: float = 0.9,
    beta: float = 3.0,
    gamma: float = 0.5,
) -> float:

    enum_hypothesis, enum_reference = _generate_enums(
        hypothesis, reference, preprocess=preprocess
    )
    translation_length = len(enum_hypothesis)
    reference_length = len(enum_reference)
    matches, _, _ = _enum_align_words(
        enum_hypothesis, enum_reference
    )
    matches_count = len(matches)
    try:
        precision = float(matches_count) / translation_length
        recall = float(matches_count) / reference_length
        fmean = (precision * recall) / (alpha * precision + (1 - alpha) * recall)
        chunk_count = float(_count_chunks(matches))
        frag_frac = chunk_count / matches_count
    except ZeroDivisionError:
        return 0.0
    penalty = gamma * frag_frac ** beta
    return (1 - penalty) * fmean



def meteor_score(
    references: Iterable[Iterable[str]],
    hypothesis: Iterable[str],
    preprocess: Callable[[str], str] = str.lower,
    alpha: float = 0.9,
    beta: float = 3.0,
    gamma: float = 0.5,
) -> float:
```

```
    return max(
        single_meteor_score(
            reference,
            hypothesis,
            preprocess=preprocess,
            alpha=alpha,
            beta=beta,
            gamma=gamma,
        )
        for reference in references
    )
```

Gambar 4.9 Kode Evaluasi menggunakan metrik METEOR.

4.4 Hasil Data Preparation

Data preparation memiliki peran penting untuk melatih model *tagger*, karena pada tahap tersebut setiap *n-gram* (penelitian ini terbatas pada 1-gram sampai 2-gram) pada kedua korpus dihitung probabilitas kemunculannya menggunakan persamaan (2) lalu diberikan peringkat persentil pada kedua gaya teks. *N-gram* yang peringkatnya memenuhi atau melampaui batas yang ditentukanlah yang akan dipakai untuk melatih model *tagger*. Pada penelitian ini, batas yang ditentukan adalah 0% atau seluruh *n-gram* yang menjadi karakteristik tiap gaya teks akan dipakai untuk melatih model *tagger*. Tabel 4.1 dan 4.2 menunjukkan 10 peringkat teratas *unigram*, sedangkan Tabel 4.3 dan 4.4 menunjukkan 10 peringkat teratas *bigram* pada korpus sopan dan tidak sopan.

Tabel 4.1 Kumpulan 10 peringkat teratas *unigram* pada korpus sopan

N-gram	***Percentile Rank (%)***
Mengapa	100
Namun	99.13
Membuat	96.8
Tahu	95.97
Tetapi	95.39
Sepertinya	95.10
Kasihan	94.24
informasinya	92.80
ditindaklanjuti	92.51
Dipakai	91.65

Tabel 4.2 Kumpulan 10 peringkat teratas *unigram* pada korpus tidak sopan

N-gram	*Percentile Rank (%)*
Kalo	100
Udah	99.67
Ko	99.34
Bener	98.35
Aja	98.68
Gitu	97.77
Rb	97.2
Dipake	96.71
Gua	96.38
Kepotong	96.05

Tabel 4.3 Kumpulan 10 peringkat teratas *bigram* pada korpus sopan

N-gram	***Percentile Rank (%)***
bagaimana ini	99.71
mengapa saya	99.42
tapi tidak	98.84
seperti itu	98.56
admin mau	98.27
xxxuserxx mengapa	97.98
ini bagaimana	97.69
saja tidak	97.40
tidak dibalas	97.12
yang lalu	96.54

Tabel 4.4 Kumpulan 10 peringkat teratas *bigram* pada korpus tidak sopan.

N-gram	***Percentile Rank (%)***
ya min	99.01
ini gimana	98.68
xxxnumberxxx rb	97.20

saya ya	95.72
dong min	93.42
min mau	92.77
tapi kok	90.14
kenapa sih	89.15
thank you	88.33
kenapa ya	87.51

Semakin tinggi peringkat n-gram pada suatu korpus maka semakin tinggi juga relevansinya terhadap gaya teksnya dan sebaliknya. Sebagai contoh, kata "kalo" pada Tabel 4.2 memiliki peringkat persentil tertinggi *unigram* pada korpus tidak sopan, hal tersebut menjadikan kata tersebut sangat relevan pada gaya teks tidak sopan namun tidak relevan sama sekali terhadap gaya teks sopan. Jika dibandingkan pada kedua korpus dari *unigram* ataupun *bigram*, kedua gaya teks tersebut memiliki sifat yang berlawanan seperti *bigram* "ini bagaimana" pada korpus sopan yang berlawanan dengan "ini gimana" pada korpus tidak sopan. Dapat dikatakan bahwa polaritas kedua gaya teks pada data sudah cukup baik.

4.5 Hasil Evaluasi Model *Tagger* dan *Generator*

Setelah melakukan pelatihan pada model *Tagger* dan *Generator* dengan iterasi 45 *epoch*, hasil evaluasi dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil evaluasi model pada masing-masing metrik.

BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR
0.605	0.475	0.385	0.318	0.573

Hasil evaluasi yang dapat dilihat pada Tabel 4.5 menunjukkan bahwa model mendapatkan performa terbaik pada **BLEU 1** dan **BLEU 2**, hal tersebut dapat dikatakan teks yang dihasilkan oleh model mendapatkan rata-rata kecocokan tertinggi pada teks yang terdapat di korpus *test* hanya pada tingkat *unigram* dan *bigram.* Hal tersebut menunjukkan bahwa model belum cukup baik untuk menghasilkan *output* yang sesuai dengan prinsip ***content preservation***.

Untuk pengukuran kefasihan atau ***fluency*** pada teks yang dihasilkan oleh model, dapat dilihat pada contoh di bawah ini:

1. ***Input***: sudah saya dm min, tolong respon

   Output: sudah saya dm admin, tolong ditindaklanjuti.

   Hasil yang diharapkan: sudah saya dm min. tolong direspon.

2. ***Input***: pagi min, ini saya top up pulsa ko gagal terus ya. itu kenapa ya min?

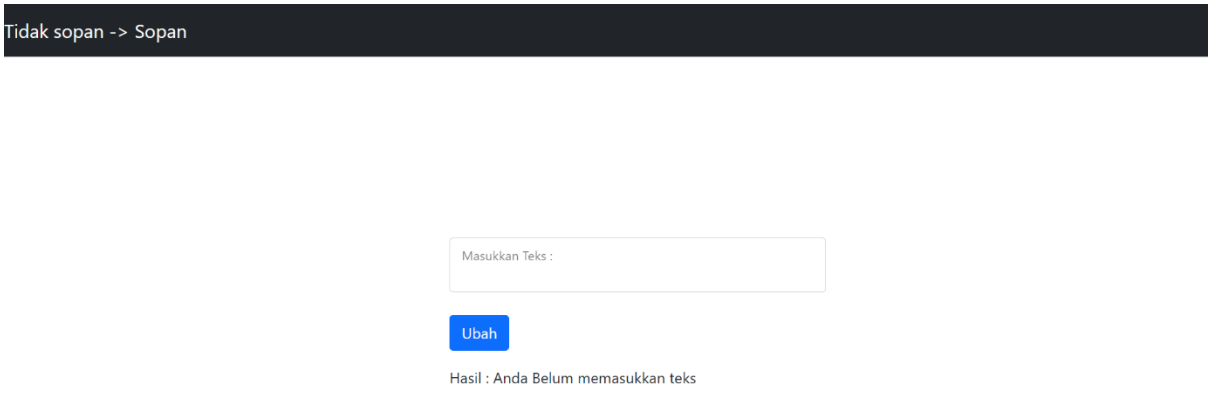
   Output: pagi admin, ini saya top up pulsa mengapa gagal terus ya. itu bagaimana?

   Hasil yang diharapkan: pagi admin. ini saya top up pulsa mengapa gagal terus? itu kenapa admin?

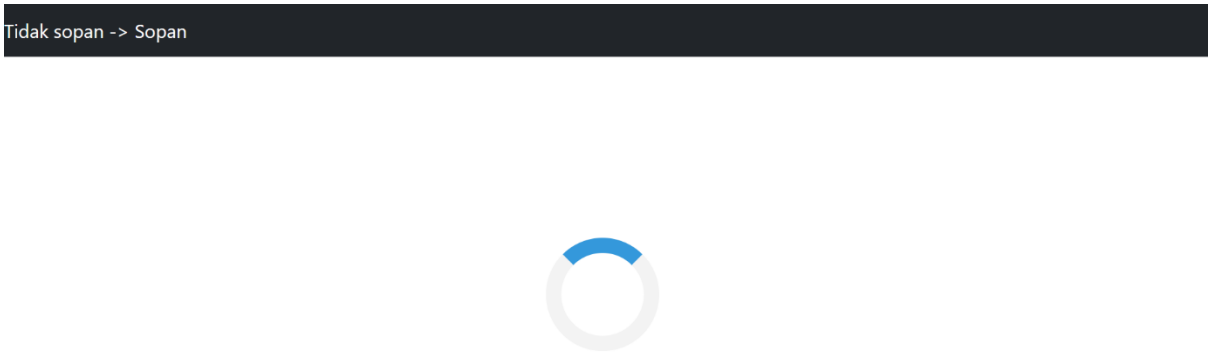
Pada kedua contoh di atas, *output* yang dihasilkan oleh model meskipun masih jauh terhadap hasil yang diharapkan namun masih dapat dipahami jika dibaca. Hal tersebut masih membutuhkan peningkatan, karena teks yang dihasilkan masih belum secara utuh berubah menjadi teks yang sopan.

4.6 User Interface

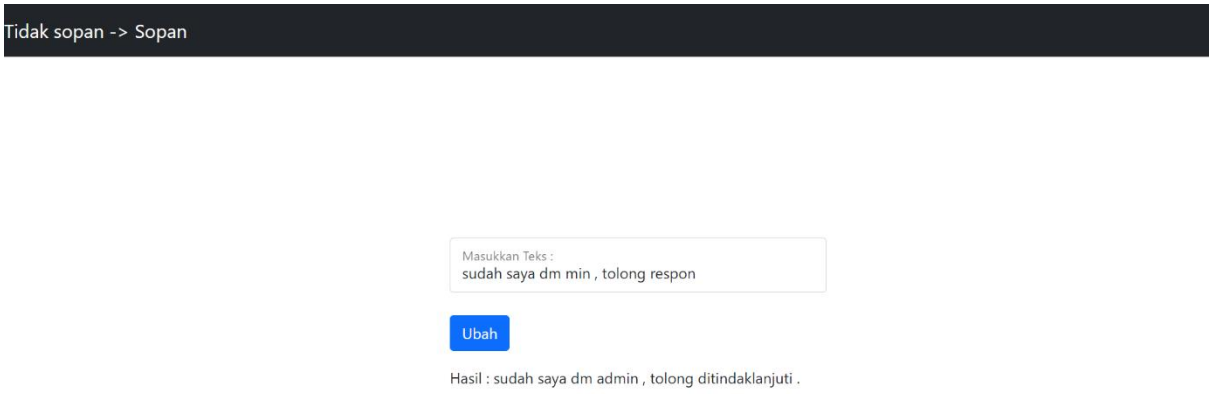
Untuk memudahkan memperlihatkan hasil model yang telah dilatih, dibuatlah sistem antarmuka menggunakan *framework flask* untuk pembuatan API dan *html*, *bootstrap*, *jquery* untuk pembuatan tampilan depan atau *front-end*. Hasil akhir dari pembuatan *User Interface* berupa sebuah halaman yang terdapat *form-input text* untuk memasukkan teks tidak sopan yang ingin diubah dan tombol ubah untuk memulai proses pengubahan oleh model seperti pada Gambar 4.10. Ketika *user* sudah memasukkan teks pada *form-input* dan menekan tombol ubah, maka *form* akan menghilang dari halaman secara sementara dan akan keluarga animasi *loader* untuk menunjukkan bahwa model sedang melakukan pengubahan teks seperti pada Gambar 4.11. Setelah model selesai melakukan pengubahan maka hasilnya akan muncul di bawah tombol seperti pada Gambar 4.12.



Gambar 4.10 Hasil *user interface* pada keaadaan awal.



Gambar 4.11 Hasil *user interface* pada keaadaan proses pengubahan teks oleh model.



Gambar 4.12 Hasil *user interface* pada keadaan model sudah selesai mengubah teks *input*.

Kode pada Gambar 4.13 adalah pengimplementasian API menggunakan bahasa pemrograman *python* dengan *framework flask*. Metode *http* yang digunakan adalah *POST* dengan *endpoint '/transform'* karena API tersebut hanya ditugaskan sebagai penerima *input*

teks yang akan diubah dan pengirim teks yang dihasilkan oleh model. Alur urutan berjalannya kode dapat dijelaskan sebagai berikut:

1. *User* melakukan *request http* dengan metode *POST* dengan *body* yang berisikan pasangan *key* bernama ‘teks’ dan *value* yang didapat dari *form-input* pada antarmuka yang tersedia.
2. Server yang berisikan kode pada Gambar 4.13 menerima *request* dari *user* dan mengambil nilai dari *key* yang bernama ‘teks’ lalu disimpan pada variabel teks.
3. Setelah berhasil mengambil teks yang akan menjadi *input* model, selanjutnya merubah teks tersebut dengan model *tagger* menggunakan fungsi *translate_sentence* dengan parameter pertama berisi “*tagger*” dan parameter kedua adalah variabel teks. Setelah model *tagger* mengeluarkan *output* berupa teks yang terdapat *token tag* P_0 hal yang harus dilakukan adalah merubahnya menjadi P_9 menggunakan fungsi *replace* dikarenakan target perubahan adalah teks sopan. Hal yang dilakukan selanjutnya adalah merubah teks yang dihasilkan oleh model *tagger* dengan model *generator* dengan memanggil fungsi *translate_sentence* menggunakan parameter “*generator*” dan variabel *tagger_output* untuk parameter kedua.
4. Tahap terakhir pada kode ini adalah mengembalikan respon kepada *user* yang berisikan *output* dari model *generator* dengan format *JSON* menggunakan bantuan fungsi *jsonify*.

```
from flask import Flask, jsonify, request
from translate import translate_sentence
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
@app.route('/transform', methods=['POST'])
def predict():
    teks = request.form['teks']
    tagger_output = translate_sentence("tagger",teks)
    tagger_output = tagger_output.replace("P_0","P_9")
    generator_output = translate_sentence("generator", tagger_output)
    return jsonify(generator_output)
```

Gambar 4.13 Kode pengimplementasian API menggunakan *framework flask.*

Kode pada Gambar 4.14 merupakan pengimplementasian tampilan antarmuka yang digunakan untuk berinteraksi dengan *user*. Untuk mempermudah pembuatan desain tampilan, kode tersebut menggunakan *Bootstrap* sebagai *library framework css.* Selain itu, kode tersebut

menggunakan *library JavaScript* bernama *Jquery* untuk membuat desain tampilan menjadi dinamis seperti membuat animasi *loader* dan mengambil hasil pengubahan model tanpa melakukan *refresh* halaman.

```
<html>
    <head>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"                                                    integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
    </head>
    <body>

        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
              <a class="navbar-brand" href="#">Tidak sopan -> Sopan</a>
        </nav>
        <div id="loader"></div>
        <div class="form-center" id="form">
            <form class="form-floating">
                <input       type="text"       id="teks"       class="form-control"
id="floatingInputValue" >
                <label for="floatingInputValue">Masukkan Teks : </label>
                <br>
                <button         type="button"         class="btn         btn-primary"
id="ubah">Ubah</button>
              </form>
              <label>Hasil :</label>
              <span id="result">Anda Belum memasukkan teks</span>

        </div>

    </body>
</html>

<style>
#loader{
    margin-left:45%;
    margin-top:15%;
    display:none;
}
.form-center {
  width:400px;
  margin-top:15%;
  margin-left:37%;
}

#loader {
  border: 16px solid #f3f3f3; /* Light grey */
  border-top: 16px solid #3498db; /* Blue */
  border-radius: 50%;
  width: 120px;
  height: 120px;
  animation: spin 2s linear infinite;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}
</style>
<script src=
"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
```

```
<link                              href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css"        rel="stylesheet"        integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"
crossorigin="anonymous">

<script>

$(document).ready(function(){
    $("#ubah").click(function(){
        let teks = $('#teks').val();
        var fd = new FormData();
        fd.append("teks",teks);
        if(teks){
            $("#form").css("display", "none");
            $("#loader").css("display", "block");
                    $.ajax({
                    url: 'http://127.0.0.1:8000/predict',
                    type: 'post',
                    data: fd,
                    contentType: false,
                    processData: false,
                    success: function(response){
                        $("#loader").css("display", "none");
                        $("#form").css("display", "block");
                        $('#result').text(response);
                    },
                });

        }else{
            alert("Masukkan Teks terlebih dahulu!");
        }
    });

});

</script>
```

Gambar 4.14 Kode tampilan depan antarmuka pengguna.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini membuat model *Text Style Transfer* untuk mengubah teks tidak sopan menjadi sopan dalam bahasa Indonesia menggunakan strategi mempertahankan bentuk formalitas pada kalimat yang disampaikan. Metode yang digunakan pada penelitian ini adalah *Tag and Generate Approach* yang secara spesifik terdapat dua model yaitu *tagger* dan *generator*, kedua model tersebut memiliki peran masing-masing terhadap teks *input.* Model *tagger* menggantikan *n-gram* yang memiliki karakteristik gaya teks tidak sopan dengan *token tag*. Sedangkan, model *generator* mengisi *token tag* dengan *n-gram* yang memiliki karakteristik gaya teks sopan.

Hasil akhir evaluasi terbaik didapatkan dengan metrik **BLEU-1** dan disusul dengan metrik **METEOR** dengan skor masing-masing 0.605 dan 0.573 dari skala 0 – 1. Hal tersebut dapat diartikan bahwa kesamaan teks yang dihasilkan oleh model dengan *output* yang diharapkan mendapatkan hasil terbaik pada *unigram* dan kesamaan semantik pada teks yang dihasilkan model. Kefasihan bahasa dari teks yang dihasilkan oleh model cukup baik untuk dipahami, namun gaya teks yang dihasilkan belum secara keseluruhan berubah. Dapat disimpulkan bahwa model pada penelitian ini dibutuhkan pengembangan lebih lanjut untuk dapat diaplikasikan di dunia nyata.

Berdasarkan pada hasil evaluasi dan *data preparation* yang didapatkan bahwa hasil penentuan *unigram* dan *bigram* yang dijadikan sebagai atribut masing-masing gaya teks belum maksimal sehingga model belum dapat mencapai hasil yang terbaik. Hal tersebut dapat dibuktikan dengan masih terdapat beberapa kata atau frasa yang menunjukkan kecenderungan terhadap gaya teks tertentu (sopan atau tidak sopan) namun tidak masuk ke dalam daftar *n-gram* yang akan menjadi pengganti *token tag* (seperti pada Tabel 4.1 sampai Tabel 4.4).

5.2 Saran

Masalah utama pada penelitian ini adalah keterbatasan jumlah data yang ada, hal tersebut berdampak pada pemilihan atribut pada tiap gaya teks yang menyebabkan kurang optimalnya pelatihan model. Oleh karena itu, disarankan pada penelitian berikutnya untuk menambah jumlah data. Selain itu, terdapat beberapa saran yang perlu dipertimbangkan seperti:

1. Cari *threshold* peringkat persentil yang terlihat jelas polaritas *n-gram* pada masing-masing gaya teks.

2. Mencoba variasi *min* dan *max* pada *n-gram range* dan pilih yang terbaik berdasarkan metrik yang berlaku.
3. Mencoba melatih model *tagger* dengan data yang diberikan *tag* secara manual atau tanpa bantuan dari peringkat persentil untuk pemilihan *n-gram*-nya.
4. Mencoba variasi arsitektur pada model *tagger* dan *generator*.

DAFTAR PUSTAKA

Abdurrahman. (2006). *Pragmatik: Konsep Dasar Memahami Konteks Tuturan*.

Brown, P. (2015). Politeness and Language. In *International Encyclopedia of the Social & Behavioral Sciences: Second Edition*. https://doi.org/10.1016/B978-0-08-097086-8.53072-4

Cao, Y., Shui, R., Pan, L., Kan, M.-Y., Liu, Z., & Chua, T.-S. (2020). *Expertise Style Transfer: A New Task Towards Better Communication between Experts and Laymen*. 1061–1071. https://doi.org/10.18653/v1/2020.acl-main.100

Dos Santos, C. N., Melnyk, I., & Padhi, I. (2018). Fighting offensive language on social media with unsupervised text style transfer. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *2*, 189–194. https://doi.org/10.18653/v1/p18-2031

Faculty, J., Studies, C., & Corresponding, I. (2014). *POLITENESS AND CAMARADERIE : HOW TYPES OF FORM MATTER IN INDONESIAN CONTEXT*. 335–350.

Hu, Z., Lee, R. K.-W., Aggarwal, C. C., & Zhang, A. (2020). *Text Style Transfer: A Review and Experimental Evaluation*. http://arxiv.org/abs/2010.12742

Jin, D., Jin, Z., Hu, Z., Vechtomova, O., & Mihalcea, R. (2020). *Deep Learning for Text Style Transfer: A Survey*. 1–47. http://arxiv.org/abs/2011.00416

Johnstone, B. (1989). Linguistic Strategies and Cultural Styles for Persuasive Discourse. In *Language, Communication and Culture* (Issue January, pp. 139–156).

Kardana, I. N., Satyawati, M. S., & Adi Rajistha, I. G. N. (2018). Strategies to Create Polite Expressions in Indonesian Communication. *International Journal of Linguistics*, *10*(6), 1. https://doi.org/10.5296/ijl.v10i6.13851

Kneser, R., & Ney, H. (1995). Improved backing-off for M-gram language modeling. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *1*, 181–184. https://doi.org/10.1109/ICASSP.1995.479394

Lavie, A., & Agarwal, A. (2007). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *Proceedings of the Second Workshop on Statistical Machine Translation*, *June*, 228–23. http://acl.ldc.upenn.edu/W/W05/W05-09.pdf#page=75

Li, J., Jia, R., He, H., & Liang, P. (2018). Delete, retrieve, generate: A simple approach to sentiment and style transfer. *NAACL HLT 2018 - 2018 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, *1*, 1865–1874. https://doi.org/10.18653/v1/n18-1169

MacArthur, C. A. (2009). Reflections on Research on Writing and Technology for Struggling Writers. *Learning Disabilities Research & Practice*, *24*(2), 93–103. https://doi.org/10.1111/j.1540-5826.2009.00283.x

Madaan, A., Setlur, A., Parekh, T., Poczos, B., Neubig, G., Yang, Y., Salakhutdinov, R., Black, A. W., & Prabhumoye, S. (2020). Politeness transfer: A tag and generate approach. In *arXiv*. https://doi.org/10.18653/v1/2020.acl-main.169

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002a). *Bleu: a Method for Automatic Evaluation of Machine Translation*. 311–318. https://doi.org/10.3115/1073083.1073135

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002b). Bleu: a Method for Automatic Evaluation of Machine Translation. *Undefined*, 311–318. https://doi.org/10.3115/1073083.1073135

Pranowo. (2012). *Berbahasa secara Santun*. Pustaka Pelajar.

Rao, S., & Tetreault, J. (2018). Dear sir or madam, may i introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 129–140. https://doi.org/10.18653/v1/n18-1012

Sudaryat, Y. (2009). *Makna dalam wacana*. Yrama Widya.

Wibowo, H. A., Prawiro, T. A., Ihsan, M., Aji, A. F., Prasojo, R. E., Mahendra, R., & Fitriany, S. (2020). Semi-Supervised Low-Resource Style Transfer of Indonesian Informal to Formal Language with Iterative Forward-Translation. *2020 International Conference on Asian Language Processing, IALP 2020*, 310–315. https://doi.org/10.1109/IALP51396.2020.9310459

Yosepha Pusparisa. (2021). *Tingkat Kesopanan Netizen Indonesia Paling Buruk Se-Asia Pasifik*. https://databoks.katadata.co.id/datapublish/2021/02/26/tingkat-kesopanan-netizen-indonesia-paling-buruk-se-asia-pasifik

Zhang, Y., Xu, J., Yang, P., & Sun, X. (2020). Learning sentiment memories for sentiment modification without parallel data. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 1103–1108. https://doi.org/10.18653/v1/d18-1138

LAMPIRAN