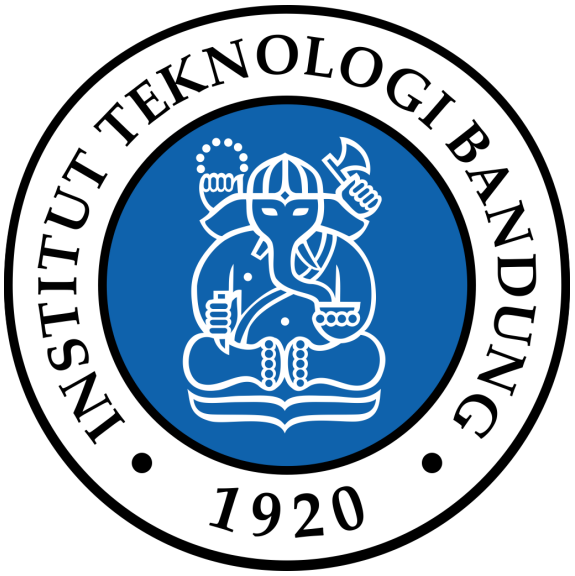
Tugas IF4072 Pemrosesan Teks dan Suara Bahasa Alami

(Teks)

Aplikasi Pendeteksi Pengaduan Masyarakat Terhadap Pemerintah berbasis Twitter

"Ada Aduan"



Disusun oleh :

Robby Syaifullah	(13515013)
Kevin Erdiza Y.	(13515016)
Patrick Nugroho H.	(13515040)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2018

A. Deskripsi Aplikasi

Aplikasi "Ada Aduan" ini merupakan aplikasi sederhana yang berjalan memanfaatkan pemrosesan bahasa natural (*Natural Language Processing*/NLP) yang berbasis teks. Aplikasi ini memproses cuitan-cuitan (*tweet-tweet*) di Twitter dan mengambil data-data yang berhubungan dengan aduan masyarakat kepada pemerintah. Data yang dianggap sebagai aduan diantaranya adalah yang memiliki kata kunci "tolong", "diperhatikan", "diproses", "presiden", dan lain-lain.

B. Latar Belakang Aplikasi (Manfaat Aplikasi)

Dewasa ini penggunaan media sosial meningkat pesat. Semua orang menggunakan media sosial untuk berinteraksi satu sama lain, baik kepada keluarga dekat mereka maupun kerabat yang ada di negara nun jauh di sana. Semua orang mengungkapkan ekspresi dan kegelisahannya melalui media sosial, seperti Twitter, Instagram, maupun Facebook. Twitter merupakan salah satu platform media sosial yang paling sering dipakai untuk mengungkapkan pendapat, terutama yang berhubungan dengan pemerintahan atau politik.

Sekarang banyak orang yang menyampaikan aduannya terhadap pemerintah melalui cuitan/*tweet* di Twitter. Twitter dirasa nyaman untuk menyampaikan keluhan karena pesan yang disampaikan singkat dan mudah disebarkan, cukup dengan *re-tweet* maupun diviralkan dengan tagar-tagar yang dapat menimbulkan fenomena "*Trending Topic*".
Keluhan terhadap pemerintah yang begitu banyak dan begitu sering membuat pemerintah kesulitan untuk menampung cuitan itu semua dan mengolahnya. Akibatnya, aspirasi masyarakat kurang tersampaikan apalagi terealisasi.

Dengan adanya aplikasi "Ada Aduan" ini pemerintah dapat lebih mudah mengontrol dan memantau aduan dari masyarakat melalui Twitter. Terkadang masalah yang diadukan tampak sepele, tetapi sebenarnya sungguh dibutuhkan oleh masyarakat ataupun berhubungan dengan khalayak banyak. Aplikasi ini membantu pemerintah lebih cepat menampung aspirasi dan aduan masyarakat, memproses, dan mengeksekusi solusinya, sehingga masalah yang dihadapi masyarakat segera teratasi. Hal ini akan menimbulkan perasaan puas (*satisfaction*) dan percaya (*trust*) pada pemerintah yang ada, sehingga kesejahteraan sosial di negara ini dapat tercapai.

C. Aplikasi Lain yang Sudah Ada (yang mirip dengan aplikasi yang diajukan)

Salah satu contoh aplikasi yang sudah ada dan mirip dengan aplikasi "Ada Aduan" adalah:

1. "*Event detection and analysis on short text messages*" Amosse Edouard, Université Côte d'Azur, 2017

   Solusi yang ditawarkan pada penelitian ini dirangkum dalam 3 tahap:

   a. *Event Classification*

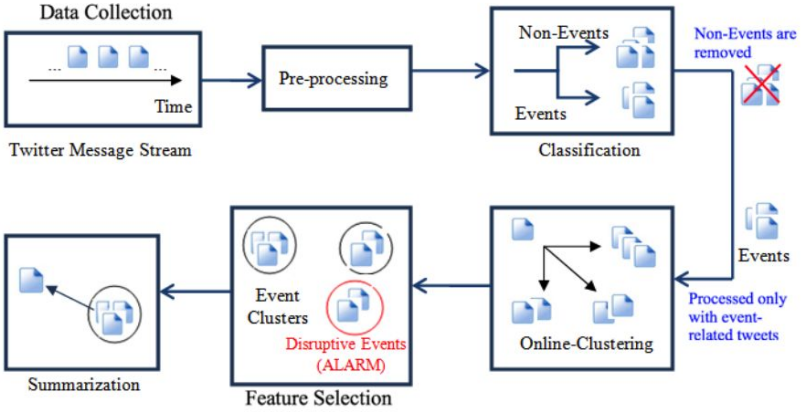
      Untuk mengklasifikasi tweet yang *event-related*, dilakukan entity linking dan generalisasi NE menggunakan DBpedia dan YAGO Ontologies. Bertujuan

untuk memisahkan *tweet* yang *event-related* dan yang tidak. Membandingkan algoritma *Naive Bayes*, *Support Vector Machine* (SVM), dan *Long Short-Term Memory* (LSTM), menunjukkan bahwa *NE linking and replacement* meningkatkan kinerja klasifikasi dan mengurangi *overfitting*, terutama dengan *Recurrent Neural Network* (RNN)

b. *Event Clustering*
   Memanfaatkan *Named Entity* (*NE) mention* dan *local context* untuk membuat *temporal event graph*. Kemudian menggunakan teknik teori graf dan algoritma *PageRank-like* untuk memproses *event graph* untuk mendeteksi kluster dari *tweet* yang menggambarkan *event* yang sama. Selain itu dapat pula mendeteksi atribut penting seperti dimana *event* tersebut terjadi, siapa orang dan organisasi yang terlibat.

c. *Event Tracking and Summary*
   Membangun *timeline* dengan aksi dari sebuah acara olahraga berdasarkan *tweet*. Hal ini dilakukan dengan mengombinasikan *external Knowledge Basis* (KB) untuk memperkaya konten dari *tweet* dan mengaplikasikan teori graf untuk memodelkan relasi antara aksi dan partisipan dalam permainan (*game*). Ini sudah diaplikasikan pada saat Euro (Kejuaraan Piala Eropa) 2016 di Paris berlangsung, lalu mengevaluasi *output* nya dengan membandingkannya dengan *live streaming* dari kanal olahraga sesungguhnya.

2. "*Event Identification in Social Media using Classification-Clustering Framework*" , Nasser Alsaedi, Cardiff University School of Computer Science & Informatics, 2017.
   Tujuan dari *framework* yang dibuat adalah mengidentifikasi *event detection* pada media sosial seperti Twitter dan Flickr. Twitter adalah sebuah situs *micro-blogging* dimana penggunanya dapat membuat pesan singkat (tidak lebih dari 140 karakter) dan mengenai topik apapun. Ada 4 tahap pada *framework* ini, yaitu *data collection*, *preprocessing*, *classification*, *clustering*, dan *summarization*.
   Tahap klasifikasi menggunakan tiga algoritma pembelajaran mesin yang berbeda untuk memisahkan *tweet* yang mengandung *event-related* dengan yang tidak. Algoritma itu diantaranya *Naive Bayes classifier*, *Logistic Regression* dan *Support Vector Machines* (SVM). Kemudian mengukur kinerjanya dengan metriks evaluasi standar.
   Menginvestigasi cara meningkatkan kinerja sistem dengan menggunakan beragam fitur seperti dengan metode n-gram atau frekuensi kemunculan, penggunaan *unigram*, *bigram* and *trigram*, fitur linguistik seperti *parts-of-speech* (POS) *tagging* dan *Named Entity Recognition* (NER).
   Tahap *clustering* bertujuan mengelompokkan *tweet* yang berisi *event-related* dalam satu *cluster* menggunakan beragam fitur, misalnya dari sisi temporal, spasial, dan tekstual, untuk meningkatkan identifikasi dari *event* tersebut, terutama *event* yang mengganggu (*disruptive event*). Tahap *summarization* berfokus pada memilih konten *tweet* dari hasil *clustering*.



Gambar 1. *Framework* Deteksi *Even*t untuk Konten Media Sosial

D. Arsitektur Aplikasi

1) Arsitektur Aplikasi

Secara umum, aplikasi memproses *tweet* pengguna Twitter dan mengklasifikasi apakah *tweet* tersebut termasuk dalam *tweet* pengaduan. Selanjutnya di-*filter* untuk mengambil *tweet* dengan label pengaduan (label "1"). *Tweet* tersebut dimasukkan ke dalam modul *POS Tagger* untuk mendapatkan *POS Tag* setiap kata dalam *tweet*. Kemudian diambil kata dengan *POS Tag* tertentu sebagai inti dari *tweet* tersebut.

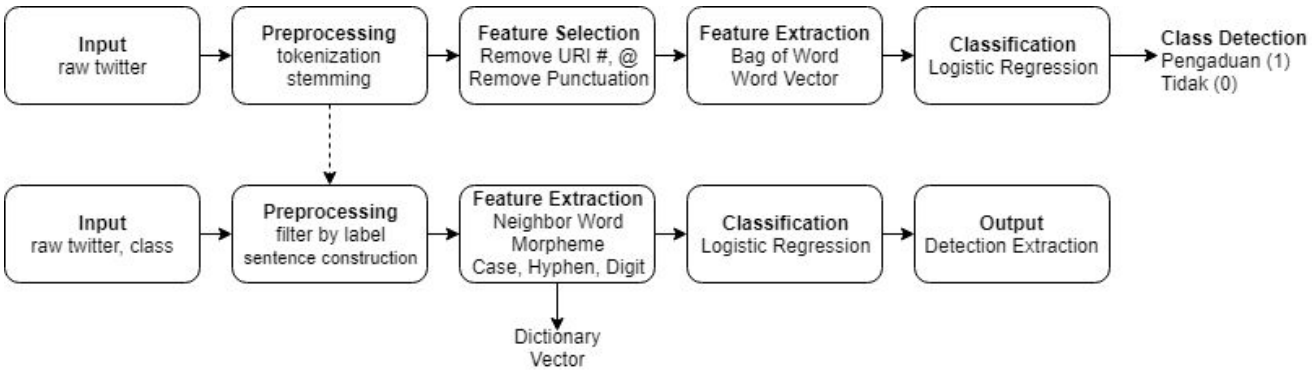
Input berupa teks *tweet* di Twitter, kemudian dilakukan *preprocessing* berupa tokenisasi dan *stemming*, agar kata-kata yang memiliki variasi *morpheme* dianggap kata yang sama. Selanjutnya dilakukan seleksi fitur yakni pembuangan *link* dari tweet, karena kata-kata pada *link* bukan fokus utama dari pemrosesan teks, dan dianggap tidak menambah kinerja dari aplikasi. Ekstraksi fitur dilakukan dengan menjadikan seluruh kata sebagai *bag of word*, lalu ditransformasi ke bentuk vektor dengan menghitung jumlah kata tersebut. Selanjutnya dilakukan pemisahan *dataset* menjadi *data training* dan *data test*.

Dengan fitur yang ada, dilakukan pelatihan dengan menggunakan pendekatan pembelajaran mesin, yakni *Logistic Regression*. Didapatkan prediksi kelas dari masukkan *dataset* berupa angka 1 dan 0. Angka 1 menunjukkan *tweet* tersebut tergolong pengaduan dan 0 sebaliknya.

Setelah itu, dilakukan penggabungan masukan dengan label prediksinya, yang akan dijadikan input baru pada modul *POS Tagger*. Pada tahap *preprocessing*, diambil data dari modul sebelumnya yakni hasil *preprocessing tokenization*. Lalu dilakukan *filter* masukan, yakni hanya mengambil data dengan label 1 dan data kembali dipecah menjadi kumpulan kata pada setiap kalimat. Setelah itu dilakukan ekstraksi fitur. Kami menggunakan fitur dasar seperti nilai/*value* dari kata saat ini dan kata tetangganya yakni satu kata sebelum dan sesudah kata tersebut. Kemudian ditambahkan fitur *morphem*, yakni *suffix* dan *prefix* dari setiap kata. Untuk bahasa Indonesia, *prefix* dari kata bisa mencapai 5 huruf, namun jarang terjadi. Umumnya

berkisar antara 2-4. Karena itu ditambahkan fitur berupa *prefix*-1, *prefix*-2, *prefix*-3, *prefix*-4, *suffix*-1, *suffix*-2, *suffix*-3. Dan juga ditambahkan pula posisi kata dalam kalimat. Dan terakhir ditambahkan fitur berupa morfologi dari kata tersebut, yakni huruf kapital di awal (*Word case*), adanya tanda hubung "-" dan membedakan kata yang terdiri dari angka saja.

Selanjutnya dilakukan vektorisasi menggunakan kakas *DictVectorizer* dari sckit sehingga semua fitur diubah menjadi vektor. Selanjutnya vektor dilatih dengan menggunakan model pembelajaran *Logistic Regression*. Sehingga menghasilkan kelas *POS Tag* dari setiap kata. Kemudian setiap kata dan *POS Tag* nya dipasangkan. Dan dimulai proses ekstraksi dari aplikasi dengan keluaran berupa : ***username*** pembuat pengaduan, **inti pengaduan** yang terdiri dari kata kerja, kata benda sebagai keterangan, dan angka/nominal; dan yang terakhir adalah ***tweet* lengkap** dari pengguna sebagai pilihan cadangan jika pembaca tidak dapat memahami maksud dari inti pengaduan. Ketiga keluaran ini dikumpulkan dan disimpan pada *file* eksternal dengan ekstensi csv. Ilustrasi arsitektur sistem ini dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur Keseluruhan Sistem

Contoh Data

1. *Input*

   `Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan pak, serta swasembada pangan. Thank's.`

2. *Preprocessing*

   (Menghapus URI, # dan @\')

   `Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan pak serta swasembada pangan Thanks`

   Tokenization

   `['Tolong', 'di', 'Kalimantan', 'Selatan', 'di', 'bantu', 'ekonomi', 'kerakyatan', 'pak', 'serta', 'swasembada', 'pangan', 'Thanks']`

   Word vector, mentransformasi kata menjadi vektor

```
  (0, 628)     1
  (0, 1179)    2
  (0, 1348)    1
  (0, 2101)    1
  (0, 3211)    1
  (0, 3668)    1
  (0, 3999)    1
  (0, 4047)    1
  (0, 4497)    1
  (0, 4586)    1
```

Kelas dari model *Text Classifier* yang sudah digabungkan dengan masukannya

```
('hairul25615019', "Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan pak, serta swasembada pangan. Thank's.", 1)
```

Ekstraksi Fitur dari masukan berupa pasangan *raw tweet* dan kelasnya

```
{'value': 'di', 'prev_word': 'Tolong', 'next_word': 'Kalimantan', 'prev_pos': 0, 'pos': 1, 'next_pos': 2, 'prefix_1': 'd', 'prefix_2': 'di', 'prefix_3': 'di', 'prefix_4': 'di', 'suffix_1': 'i', 'suffix_2': 'di', 'suffix_3': 'di', 'is_capitalized': False, 'has_hyphen': False, 'is_digit': False}
```

Hasil dari *POS Tag* yang telah digabung menjadi keluaran siap di-*export* menjadi *file* eksternal

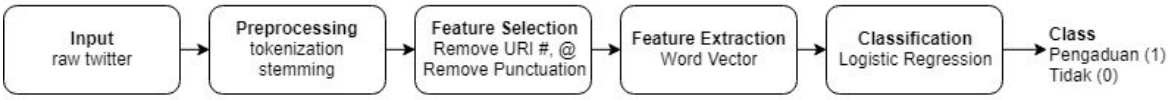
```
{'username': 'hairul25615019', 'Tindakan': [], 'Nominal': [], 'Keterangan': ['Tolong', 'Kalimantan', 'Selatan', 'bantu', 'ekonomi', 'kerakyatan', 'pak', 'pangan', 'Thanks'], 'tweet_lengkap': "Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan pak, serta swasembada pangan. Thank's."}
```

2) ***Modul Training Text Classification***

Klasifikasi teks memiliki arsitektur yang sama dengan arsitektur umum di atas. Secara umum, aplikasi memproses *tweet* pengguna Twitter dan mengklasifikasi apakah *tweet* tersebut termasuk dalam *tweet* pengaduan. Selanjutnya di-*filter* untuk mengambil *tweet* dengan label pengaduan (label "1"). *Tweet* tersebut dimasukkan ke dalam modul *POS Tagger* untuk mendapatkan POS Tag setiap kata dalam *tweet*. Kemudian diambil kata dengan *POS Tag* tertentu sebagai inti dari *tweet* tersebut.

Input berupa teks *tweet* twitter, kemudian dilakukan *preprocessing* berupa tokenisasi dan *stemming*, agar kata-kata yang memiliki variasi *morpheme* dianggap kata yang sama. Selanjutnya dilakukan seleksi fitur yakni pembuangan *link* dari tweet, karena kata-kata pada *link* bukan fokus utama dari pemrosesan teks, dan dianggap tidak menambah kinerja dari aplikasi. Ekstraksi fitur dilakukan dengan menjadikan seluruh kata sebagai *bag of word*, lalu ditransformasi ke bentuk vektor dengan menghitung jumlah kata tersebut. Selanjutnya dilakukan pemisahan *dataset* menjadi *data training* dan *data test*.

Dengan fitur yang ada, dilakukan pelatihan dengan menggunakan pendekatan pembelajaran mesin, yakni *Logistic Regression*. Didapatkan prediksi kelas dari masukkan *dataset* berupa angka 1 dan 0. Angka 1 menunjukkan *tweet* tersebut tergolong pengaduan dan 0 sebaliknya. Ilustrasi dapat dilihat pada gambar 3.



Gambar 3. Arsitektur Modul Klasifikasi Teks

Contoh Data pada Modul Klasifikasi Teks

1. *Input*

   ```
   Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan
   pak, serta swasembada pangan. Thank's.
   ```

2. (Menghapus URI, # dan @\')

   ```
   Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan
   pak serta swasembada pangan Thanks
   ```

2. Tokenization

   ```
   ['Tolong', 'di', 'Kalimantan', 'Selatan', 'di', 'bantu',
   'ekonomi', 'kerakyatan', 'pak', 'serta', 'swasembada',
   'pangan', 'Thanks']
   ```

3. Word vector, mentransformasi kata menjadi vektor

   ```
   (0, 628)     1
   (0, 1179)    2
   (0, 1348)    1
   (0, 2101)    1
   (0, 3211)    1
   (0, 3668)    1
   (0, 3999)    1
   (0, 4047)    1
   ```

```
  (0, 4497)     1
  (0, 4586)     1
```

4. Kelas dari model Text Classifier yang sudah digabungkan dengan masukannya
   `('hairul25615019', "Tolong di Kalimantan Selatan di bantu ekonomi kerakyatan pak, serta swasembada pangan. Thank's.", 1)`

3) **Modul *POS Tag* untuk *training model POS Tag***

Input modul ini berupa kalimat yang berbentuk *list* kata beserta *POS Tag* masing-masing kata. *Preprocessing* yang dilakukan cukup menggabungkan kata dengan labelnya. Setelah itu dilakukan ekstraksi fitur. Kami menggunakan fitur dasar seperti nilai/*value* dari kata saat ini dan kata tetangganya yakni satu kata sebelum dan sesudah kata saat ini,. Kemudian ditambahkan fitur *morphem*, yakni *suffix* dan *prefix* dari setiap kata. Untuk bahasa Indonesia, *prefix* dari kata bisa mencapai 5 huruf, namun jarang terjadi. Umumnya berkisar antara 2-4. Karena itu ditambahkan fitur berupa *prefix*-1, *prefix*-2, *prefix*-3, *prefix*-4, *suffix*-1, *suffix*-2, *suffix*-3. Dan juga ditambahkan posisi kata dalam kalimat. Dan terakhir ditambahkan fitur berupa morfologi dari kata tersebut. Yakni huruf kapital di awal (*Word case*), adanya tanda hubung "-" dan membedakan kata yang terdiri dari angka saja.

Selanjutnya dilakukan vektorisasi menggunakan kakas DictVectorizer dari sckit sehingga semua fitur telah diubah menjadi vektor. Selanjutnya vektor dilatih dengan menggunakan model pembelajaran *Logistic Regression*. Sehingga menghasilkan kelas POS Tag dari setiap kata. Kemudian setiap kata dan *POS Tag* nya dipasangkan. Dan dimulai proses ekstraksi dari aplikasi dengan keluaran berupa : ***username*** pembuat pengaduan, **inti pengaduan** yang terdiri dari kata kerja, kata benda sebagai keterangan, dan angka/nominal; dan yang terakhir adalah ***tweet* lengkap** dari pengguna sebagai pilihan cadangan jika pembaca tidak dapat memahami maksud dari inti pengaduan. Ketiga keluaran ini dikumpulkan dan disimpan pada *file* eksternal dengan ekstensi csv. Ilustrasi arsitektur ini dapat dilihat pada Gambar 4.

Gambar 4. Arsitektur Modul POS Tag

Contoh *input* data pada modul *POS Tagger*

1. *Input*
   `['Kera\tNN', 'untuk\tSC', 'amankan\tVB', 'pesta olahraga\tNN', '']`

2. *Preprocessing* (membuat pairing instance label)

```
[['Kera', 'NN'], ['untuk', 'SC'], ['amankan', 'VB'],
['pesta olahraga', 'NN']]
```

3. Ekstraksi Fitur

```
{'value': 'untuk', 'prev_word': 'Kera', 'next_word':
'amankan', 'prev_pos': 0, 'pos': 1, 'next_pos': 2,
'prefix_1': 'u', 'prefix_2': 'un', 'prefix_3': 'unt',
'prefix_4': 'untu', 'suffix_1': 'k', 'suffix_2': 'uk',
'suffix_3': 'tuk', 'is_capitalized': False, 'has_hyphen':
False, 'is_digit': False}
```

4. *Output*

```
[Kera, NN]
```

E. Teknik yang Digunakan

1. Logistic Regression

Logistic Regression merupakan salah satu model statistik yang menggunakan fungsi logistik untuk memprediksi nilai yang bertipe kategorikal dari nilai prediktor yang bersifat kontinu atau kategorikal. Dalam aplikasi "Ada Aduan", *Logistic Regression* tepat digunakan dalam proses identifikasi apakah suatu *tweet* merupakan aduan atau tidak karena permasalahan tersebut merupakan permasalahan klasifikasi biner (kelas [pengaduan(1), tidak (0)] dengan nilai prediktor yang kategorikal (*word vector*). *Logistic Regression* juga tepat digunakan pada proses *training model POS tag* karena merupakan proses klasifikasi ke sejumlah kategori (sejumlah *POS tag*) dari nilai prediktor yang kategorikal (berbagai fitur kata yang merepresentasikan konteks kata pada kalimat (posisi, kata yang bersebelahan, *prefix*, *suffix*, dsb)

F. Eksperimen

Bagian eksperimen mencakup informasi data, skenario eksperimen, hasil eksperimen dan analisisnya.

1. Informasi Data

Untuk *input dataset* dari Twitter, dilakukan *crawling* pada situs Twitter dengan menggunakan kakas twitterscraper (https://github.com/taspinar/twitterscraper). Dengan pencarian *tweet* dibatasi pada *keyword* "Presiden, Pemerintah, Menteri, atau Jokowi" dan dikombinasikan dengan kata "Tolong atau Mohon". Kami mengasumsikan mayoritas pengaduan yang ditulis masyarakat akan melibatkan 4 entitas pertama tersebut, dan pola dari *tweet* pengaduan seringkali menggunakan kata "mohon" atau "tolong" yang mengindikasikan adanya masalah yang butuh ditangani. Data yang diambil dipilih secara acak dalam jangka waktu mulai dari 1 Januari 2017. Selanjutnya didapatkan 1187 data, dilakukan pembersihan karena ada beberapa duplikasi pada data. Sehingga data *training* akhirnya berjumlah 1135 data . Setelah itu

dilabeli manual dengan perbandingan label 1 dan 0 pada keseluruhan data adalah 4:11. Berikut *query* yang dijalankan pada kakas tersebut.

```
“twitterscraper “(Pemerintah OR Jokowi OR Menteri OR Presiden) AND (tolong OR mohon)” -l 100 -bd 2017-01-01 -ed 2018-11-11 -o tweets.json”
```

Untuk *dataset POS Tag*, data diambil dari *training POS Tag* bahasa Indonesia, yang telah tersedia pada https://github.com/famrashel/idn-tagged-corpus. Data terdiri kalimat-kalimat yang tersusun dari kata dan POS Tag untuk kata tersebut. Jumlah data dari korpus ini adalah 266651 *instance.*

2. Desain Eksperimen

a. Pada *POS Tag*

1) Variasi Jumlah *Training Data*

Dalam pembelajaran data latih, dilakukan beberapa eksperimen dengan menggunakan jumlah data yang berbeda. Kemudian dianalisis pengaruh jumlah data terhadap kinerja model. Jumlah *train dataset* untuk pelatihan divariasikan dengan jumlah berikut:

100, 500, 1000, 5000, 10000, dan 20000

Berikut adalah tabel yang memetakan nilai hasil eksperimen untuk setiap jumlah *train dataset*:

Tabel 1. Nilai Hasil Eksperimen Setiap Jumlah *Training Dataset*

Jumlah *Train Dataset*	*F1 Score*	*Accuracy Score*	*Precision Score*	*Recall Score*
100	0.591	0.610	0.417	0.817
500	0.731	0.764	0.537	0.850
1000	0.833	0.842	0.634	0.914
5000	0.854	0.914	0.709	0.953
10000	0.880	0.936	0.770	0.940
20000	0.879	0.947	0.809	0.951

Analisis

Dilihat dari hasil eksperimen, terjadi peningkatan signifikan dari jumlah dataset 100 ke 5.000, ini dikarenakan kelas dari data yang banyak (*multiclass*) sehingga semakin banyak data, akan meningkatkan kinerja.

Namun pada saat data lumayan besar yakni 10.000. *Recall* dari data menurun dan *Precision* tetap meningkat, ini berarti semakin banyak data yang diklasifikasi sesuai dengan kelas aslinya.

2) Variasi Fitur

Kami menggunakan *baseline* berupa fitur dasar seperti nilai/*value* dari kata saat ini dan tetangganya yakni satu kata sebelum dan sesudah kata saat ini, Fitur ini dilabeli FT1. Kemudian ditambahkan fitur morphem, yakni *suffix* dan *prefix* dari setiap kata. Untuk bahasa Indonesia, *prefix* dari kata bisa mencapai 5 huruf, namun jarang terjadi. Umumnya berkisar antara 2-4. Karena itu ditambahkan fitur berupa *prefix*-1, *prefix*-2, *prefix*-3, *prefix*-4, *suffix*-1, *suffix*-2, *suffix*-3. Dan juga ditambahkan posisi kata dalam kalimat. Fitur ini dinamakan FT2. Dan terakhir ditambahkan fitur berupa morfologi dari kata tersebut. Yakni huruf kapital di awal (*Word case*), adanya tanda hubung "-" dan membedakan kata yang terdiri dari angka saja. Fitur ini dinamakan FT3

Berikut adalah tabel yang memetakan nilai hasil eksperimen untuk setiap variasi fitur *dataset*:

Tabel 2. Nilai Hasil Eksperimen Setiap Variasi Fitur *Dataset*

Variasi Fitur	*F1 Score*	*Accuracy Score*	*Precision Score*	*Recall Score*
Value + Neighbor Word (FT1)	0.757	0.779	0.545	0.939
FT1 + position +morphemes (FT2)	0.857	0.914	0.712	0.944
FT2 + Word case, hyphen, dan digit (FT3)	0.854	0.914	0.709	0.953

Analisis

Fitur pertama memberikan nilai yang cukup rendah, dan peningkatan signifikan terjadi antara FT 1 dan FT2, ini menunjukkan bahwa pengaruh *morpheme* dan posisi dari kata dalam kalimat sangat mempengaruhi klasifikasi *POS Tag* kata tersebut.

b. Pada *Text Classification*

Dikarenakan keterbatasan dari jumlah *training data* dan parameter yang sederhana, dilakukan eksperimen hanya pada variasi model *learning*. Yakni melakukan pelatihan dengan tiga model yang umum digunakan dalam pengklasifikasian data. Yakni *Logistic Regression* (LR), *Decision Tree Learning* (DTL) dan *K-Nearest Neighbor* (KNN).

Variasi *Model Learning*

Berikut adalah tabel yang memetakan nilai hasil eksperimen untuk setiap variasi *model training*:

Tabel 3. Nilai Hasil Eksperimen Setiap Variasi *Model Training*

Variasi *Model Learning*	*F1 Score*	*Accuracy Score*	*Precision Score*	*Recall Score*
LR	0.481	0.711	0.400	0.603
DTL	0.410	0.655	0.358	0.479
K-NN	0.147	0.673	0.084	0.571

Keterangan: LR = *Logistic Regression*, DTL = *Decision Tree Learning*, K-NN = *K-Nearest Neighbor*

Analisis

Pada *Logistic Regression*, pelabelan *instance* berdasarkan bobot dari fitur yang dimilikinya (*logistic loss*), dan juga dengan algoritma pengklasifikasiannya yang memiliki keunggulan pada *bias-variance tradeoff* dan *Logistic Regression* akan sangat bagus untuk digunakan pada *dataset* dengan jumlah *noise* sedikit seperti yang ada pada tugas ini. Sehingga, nilai akurasi dan F1nya relatif tinggi dan paling tinggi diantara keempat algoritma yang dipakai pada tugas ini.

Pada KNN, pelabelan *instance* hanya berdasarkan faktor jarak antar fitur, tanpa memperhitungkan bobotnya. Sehingga, nilai F1 dan akurasinya lebih rendah daripada algoritma pemodelan lainnya.

Pada DTL, ada perhitungan bobot dalam melakukan klasifikasi, yang disebut *information gain*. Namun DTL tidak lebih baik daripada *Logistic Regression* karena ketika *base-rate* rendah DTL akan mengalami masalah dalam melakukan *split*, atau bahkan tidak bisa melakukan *split*. Dikarenakan tidak adanya algoritma yang sempurna untuk menyelesaikan masalah klasifikasi, maka akan selalu ada kesalahan dalam proses klasifikasi *email spam* tersebut.

G. Penjelasan peran setiap anggota tim dalam membangun aplikasi dan melakukan eksperimen

NIM	Nama	Uraian Tugas
13515013	Robby Syaifullah	Implementasi Modul *POS Tag*, Penggabungan semua modul, Eksperimen *Text Classification*, Laporan

13515016	Kevin Erdiza Y.	Implementasi Modul *Text Classification*, Pelabelan Data
13515040	Patrick Nugroho H.	Eksperimen *POS Tag*, Pelabelan Data, Laporan