LAMPIRAN – LAMPIRAN

Lampiran 1. *Source Code Preprocessing Text*

```
# Preprocessing Step
def checkEmoji(self, word):
     def msg_space(): return (re.sub(r'(\S)\\', r'\1 \\', obj)
                              for obj in word())
     def msg():
         return (nltk.word_tokenize(re.sub('(?<=&)amp|(?<=&)gt', '
', obj)) for obj in msg_space())
     def re_msg(): return ([obj for obj in row if not
                    obj.startswith('\\')] for row in msg())
     def result(): return (TreebankWordDetokenizer().detokenize(
            [new_obj for new_obj in obj]) for obj in re_msg())

     return result

def caseFoldingAndPurify(self, word):
     return re.sub(r'|'.join(self.to_match), ' ', word.lower())

def tokenizing(self, word):
     def result(): return
(nltk.word_tokenize(self.caseFoldingAndPurify(obj)) for obj in
word())
     return result

def replacing(self, tokenized, slang, abusive):
     def iterate_tokenized():
         for obj in tokenized():
            def replaced(): return iterate_condition_slang(obj)
                yield replaced

     def iterate_condition_slang(obj):
         for word in obj:
             found = False
             for row in slang:
                if row[1] == word:
                   found = True
                   if row[2] in abusive():
```

```
                        yield word
                    else:
                        yield row[2]
                if not found:
                    yield word

    def result(): return iterate_tokenized()
    return result

def filtering(self, replaced):
    def result(): return
(self.stopword.remove(TreebankWordDetokenizer().detokenize((new_obj
for new_obj in obj()))) for obj in replaced())

    return self.tokenizing(result)

def stemming(self, filtered):
    def result(): return
(self.stemmer.stem(TreebankWordDetokenizer().detokenize(obj)) for
obj in filtered())

    return self.tokenizing(result)
```

Lampiran 2. *Source Code* Pemecahan Bentuk *N-Gram*

```
# Preprocessing Step
def getBigramTrigramList(self, n, data_abusive):
        if data_abusive:
            abusive = (data for data in data_abusive())
            self.word_dataset = (
                set(chain(abusive, self.word_dataset)))

        def iterate_word_dataset():
            for row in self.word_dataset:
                padded_row = "_%s_" % (row)
                target = [padded_row[x:x+n]
                          for x in range(len(padded_row)-n+1)]
                yield target

        return iterate_word_dataset()
```

Lampiran 3. *Source Code* Pemodelan *N-Gram* Perhitungan Probabilitas Kata

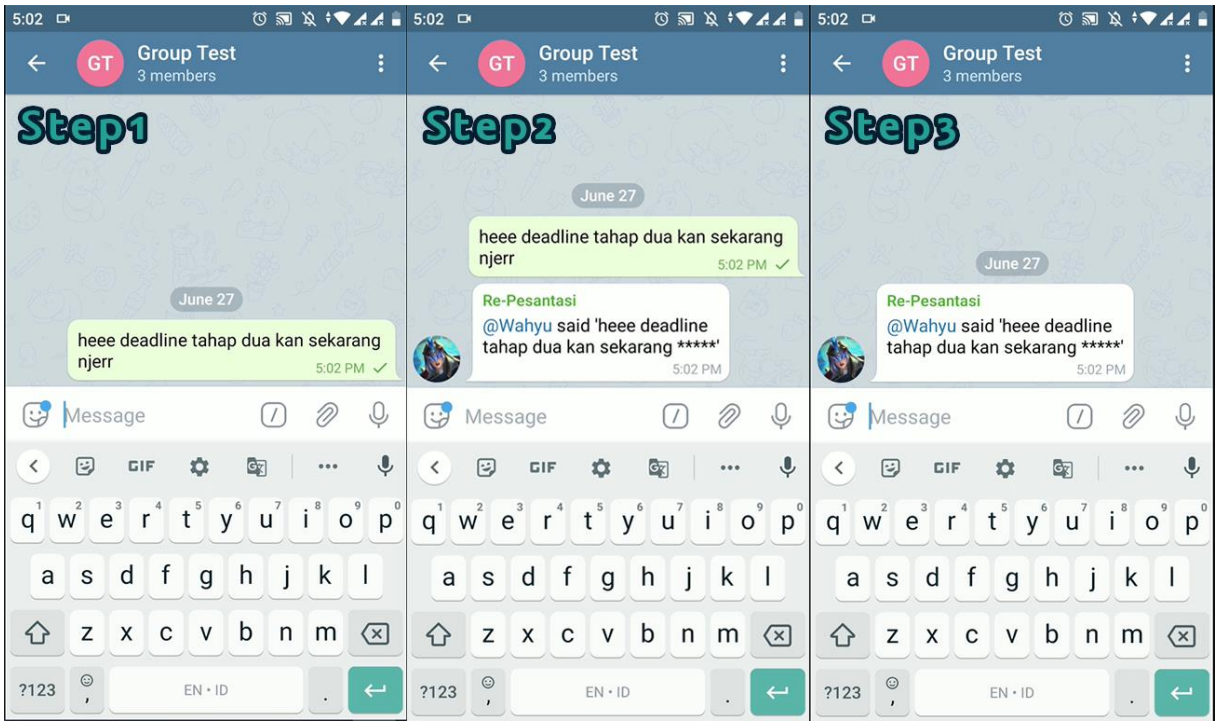
```
train_data, padded_word = padded_everygram_pipeline(n,
list(ngram_list))

# # Lets train a n-grams maximum likelihood estimation model.
model = MLE(n)
model.fit(train_data, padded_word)

ngram_prob = {}
for row in self.word_dataset:
    padded_row = "_%s_" % (row)
    prob = 0
    for x in range(len(padded_row)-n):
        word = padded_row[x:x+n]
        next_word = padded_row[(x+1):(x+1)+n]
        try:
            prob += float(model.counts[[word]] next_word] /
                          model.counts[word])
        except ZeroDivisionError:
            prob += 0
    if row in data_abusive():
        ngram_prob[row] = prob
```

Lampiran 4. *Source Code* Perhitungan Persamaan *N-Gram*

```
def count_similarity_word(row, prob, duplicate_stem,
duplicate_replace):
     row_unstem = row
     while check_abusive_slang(row_unstem):
     if mode == "admin":
        index_label = get_index_label(row, duplicate_stem)

     label = database.get_per_label_abusive(self.mark_word[0])
     score = (prob - self.mark_word[1])**2 / self.mark_word[1]

     if check_stemmer_word(row, duplicate_replace):
        row_unstem = check_stemmer_word(row, duplicate_replace)

     if label == 1:
         self.is_deleted_msg = True
```

```
            if score < threshold:
                if mode == "admin":
                    if self.label_prediksi[index_label] == 2:
                        self.label_prediksi[index_label] = 3
                    else:
                        self.label_prediksi[index_label] = label

                self.replaced_result = [[re.sub('(?<![a-zA-
Z])%s(?![a-zA-Z])' % (row_unstem), create_star(row_unstem),
word) for word in row_replace] for row_replace in
self.replaced_result]
            else:
                if mode == "admin":
                    self.label_prediksi[index_label] = 0

                self.replaced_result = [[re.sub('(?<![a-zA-
Z])%s(?![a-zA-Z])' % (row_unstem),
create_mark_label1(row_unstem), word) for word in row_replace]
for row_replace in self.replaced_result]
```

Lampiran 5. Contoh Hasil Uji Coba Bot pada Group Chat



Lampiran 6. Form Verifikasi Abstrak dan Tata Cara Penulisan Laporan Skripsi

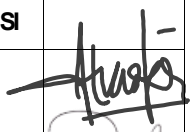
KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI TEKNIK INFORMATIKA
JL. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122

No. Skripsi : 605

FORM VERIFIKASI

ABSTRAK BAHASA INGGRIS DAN TATA TULIS BUKU LAPORAN SKRIPSI

Nama Mahasiswa : Wahyu Hidayah **NIM** : 1741720190
Tanggal Ujian : 21 Juli 2021
Judul : Aplikasi Filter Pesan Menggunakan Metode N-Gram Berbasis Telegram Bot

NO	BAGIAN YANG DIVERIFIKASI	NAMA VERIFIKATOR	TANGGAL VERIFIKASI	TTD
1	Abstrak Berbahasa Inggris	Atiqah Nurul Asri, S.Pd., M.Pd.	26 Agustus 2021	
2	Tata Tulis Buku Laporan Skripsi	Wilda Imama Sabilla, S.Kom., M.Kom	28 Juni 2021	

FRM.RIF.01.46.01