# **LAPORAN PROYEK PENGOLAHAN DATA BESAR**

**Sentiment Analysis   
Pada Streaming Data Twitter**

****

**Disusun Oleh:**

|  |  |
| --- | --- |
| **12S18033** | **Cristina Sriwahyuni Hasibuan** |
| **12S18038** | **Naomi A. Simatupang** |
| **12S18049** | **Natasya Sitorus** |
| **12S18060** | **Elsaday Bakara** |

**PROGRAM STUDI SARJANA SISTEM INFORMASI**

**FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO**

**INSTITUT TEKNOLOGI DEL**

**2022**

# **DAFTAR ISI**

[**LAPORAN PROYEK PENGOLAHAN DATA BESAR** 1](#_Toc105330299)

[**DAFTAR ISI** 2](#_Toc105330300)

[**DAFTAR GAMBAR** 3](#_Toc105330301)

[**DAFTAR KODE** 4](#_Toc105330302)

[**PENDAHULUAN** 5](#_Toc105330303)

[1.1 Latar belakang 5](#_Toc105330304)

[1.2 Tujuan 6](#_Toc105330305)

[1.3 Manfaat 6](#_Toc105330306)

[1.4 Ruang Lingkup 6](#_Toc105330307)

[**PERMASALAHAN** 7](#_Toc105330308)

[**SOLUSI** 8](#_Toc105330309)

[**MACHINE LEARNING PIPELINE** 9](#_Toc105330310)

[**IMPLEMENTASI** 10](#_Toc105330311)

[**HASIL** 17](#_Toc105330312)

[**KESIMPULAN** 18](#_Toc105330313)

[**REFRENSI** 19](#_Toc105330314)

# **DAFTAR GAMBAR**

[Gambar 1 Arsitektur 7](#_Toc105329897)

[Gambar 2 Machine learning pipeline 8](#_Toc105329898)

# **DAFTAR KODE**

[Kode 1 Library yang digunakan 10](#_Toc105329917)

[Kode 2 Start a session 10](#_Toc105329918)

[Kode 3 Configuration 11](#_Toc105329919)

[Kode 4 Set up 11](#_Toc105329920)

[Kode 5 Scrapping 11](#_Toc105329921)

[Kode 6 Load raw data twitter 11](#_Toc105329922)

[Kode 7 Cleaning tweets 12](#_Toc105329923)

[Kode 8 Labeling data 13](#_Toc105329924)

[Kode 9 Dataframe Pyspark SQL 13](#_Toc105329925)

[Kode 10 Split train test 13](#_Toc105329926)

[Kode 11 HashingTF + IDF + Logistic Regression 14](#_Toc105329927)

[Kode 12 CountVectorizer + IDF + Logistic Regression 14](#_Toc105329928)

[Kode 13 Ngram 15](#_Toc105329929)

[Kode 14 N Gram Without Chi Squared Feature Selection 16](#_Toc105329930)

[Kode 15 Visualisasi Data 16](#_Toc105329931)

# **PENDAHULUAN**

Pada bab ini membahas mengenai latar belakang, tujuan, dan ruang lingkup dari pengerjaan proyek yang akan dilakukan.

## 1.1 Latar belakang

Wabah penyakit yang disebabkan oleh virus korona (2019-nCoV) atau biasa kita sebut dengan COVID-19 telah ditetapkan sebagai pandemi global oleh WHO pada tanggal 11 Maret 2020 [1]. Meskipun saat ini pasien yang terinfeksi oleh virus tersebut mulai menurun namun masih menarik perhatian masyarakat karna hingga saat ini pandemi tersebut memberi dampak pada setiap aspek kehidupan masyarakat. Meskipun saat ini kebijakan yang berlaku dimasa pandemi mulai berubah seperti kebijakan terkait pemakaian masker, karantina, pemeriksaan kesehatan dan lainnya tetap saja akan ada efek positif maupun negatif dari pandemi ini yang dirasakan oleh masyarakat. Dalam melakukan peralihan kebijakan dari masa pandemi menuju masa normal tentunya perlu mempertimbangkan bagaimana pendapat masyarakat berdasarkan dampak positif dan negatif maupun netral yang masyarakat rasakan.

Saat ini masyarakat dapat dengan mudah menyampaikan opini dan respon mereka terhadap sesuatu baik itu dalam konteks negatif atau positif, opini dan respon ini dapat kita temukan di berbagai media sosial. Kini media sosial seolah menjadi suatu hal yang wajib dimiliki oleh seluruh masyarakat. Khususnya di Indonesia, media sosial yang banyak digunakan oleh masyarakat adalah twitter dengan total 52% dari total pengguna media sosial di Indonesia [2]. Dengan jumlah pengguna yang cukup besar ini dapat disimpulkan bahwa terdapat sumber data yang cukup besar yang dapat dimanfaatkan untuk menghasilkan informasi yang diolah menjadi knowledge yang bermanfaat.

Dengan memanfaatkan data yang bersumber dari media sosial tentunya dapat dijadikan sebagai alternatif untuk menggantikan survei tradisional. Dalam melakukan pengumpulan data respon dan opini secara tradisional baik itu secara survei atau wawancara secara langsung/real time dinilai sulit dilakukan karena adanya proses yang perlu dilakukan sehingga akan memakan waktu yang cukup lama. Terutama jika mengingat Indonesia adalah negara berkembang dengan wilayah yang luas dan penduduk yang sangat banyak mengakibatkan sulitnya untuk mendapatkan respon dan opini publik secara langsung.

Analisis sentimen atau opinion mining merupakan cabang penelitian text mining yang bertujuan untuk menentukan persepsi atau subjektivitas publik terhadap suatu topik pebahasan, kejadian atau permasalahan [3]. Analisis sentimen melakukan klasifikasi teks ke dalam orientasi positif atau negatif, analisis sentimen dibagi menjadi empat jenis pendekatan yaitu Machine learning approach, Lexicon-bassed approach, Rule-based approach dan Statistical model approach.

Pada penelitian ini akan mencari tahu performa klasifikasi dengan Logistic Regression dan Ngram. Kami menggunakan dua algoritma ini untuk membandingkan algoritma manakah yang memiliki akurasi yang lebih baik dalam melakukan klasifikasi sentimen. Logistic Regression adalah teknik klasifikasi yang digunakan ketika nilai variabel target bersifat kategoris, teknik ini sering digunakan ketika data memiliki keluaran biner. N-grams adalah potongan N-karakter dari sebuah string yang mencakup karakter apapun yang ada dalam sebuah kata yang digunakan untuk pengenalan bahasa. Karena teknik ini melakukan penguraian string menjadi bagian yang lebih kecil sehingga sesuai untuk menangani kesalahan tekstual dan pengenalan karakter dalam dokumen.

Hasil dari penelitian ini adalah grafik dari klasifikasi yang menampilkan perbandingan sentimen positif, negatif dan netral yang dirasakan oleh masyarakat terhadap pandemi covid melalui data yang berasal dari twitter dalam bahasa Indonesia. Grafik perbandingan ini diharapkan dapat menjadi pertimbangan bagi pihak yang akan membutuhkan hasil penelitian kami.

## 1.2 Tujuan

Adapun tujuan dari penelitian ini adalah untuk melakukan analisis sentimen terkait opini masyarakat terkait pandemi covid yang terjadi selama ini baik itu dalam orientasi positif dan negatif.

## 1.3 Manfaat

Manfaat dari pengerjaan proyek ini yaitu:

1. Bagi Mahasiswa

Proyek ini diharapkan nantinya dapat memberikan pemahaman dan keterampilan bagi tim proyek untuk mengimplementasikan, mengintegrasikan, dan mengimplementasikan teknologi yang digunakan dalam pengolahan data besar.

## 1.4 Ruang Lingkup

Ruang Lingkup dalam pengerjaan proyek ini adalah

1. Menggunakan data dari API twitter.
2. Menggunakan Google Colab sebagai *worksheet* untuk menyimpan, menulis, dan membagikan program yang dibagun.
3. Analisis sentimen akan dilakukan dengan klasifikasi opini masyarakat berbahasa Indonesia yang dikumpulkan dari twitter dengan menggunakan *Logistic Regression* dan *N-gram*.

# **PERMASALAHAN**

Pada bab ini dijelaskan masalah yang akan diselesaikan dalam proyek Pengenalan Big Data.

Permasalahan proyek pengenalan Big Data yang akan diselesaikan adalah melakukan klasifikasi analisis sentimen pada pendapat masyarakat terkait pandemi covid. Arsitektur sistem akan didesain dapat mengolah data terstruktur. Hal tersebut berguna untuk mengetahui cara klasifikasi yang dapat digunakan apabila data yang digunakan merupakan data streaming. Berdasarkan kebutuhan diatas maka permasalahan yang akan dirumuskan adalah bagaimana suatu sistem dapat mengklasifikasikan opini yang diberikan oleh masyarakat terkait pandemi covid yang terjadi yang bernilai positif, netral, atau negatif.

# **SOLUSI**

**2.1 Big Data Architecture**

Berikut merupakan arsitektur big data yang menjadi solusi terhadap permasalahan untuk mengklasifikasi setiap opini yang diberikan oleh masyarakat terkait pandemi Covid sehingga menghasilkan klasifikasi sentimen yang positif, negatif dan netral.



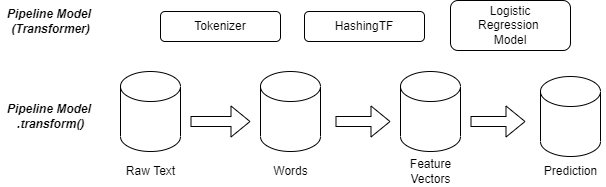
Gambar 1 Arsitektur

Arsitektur sistem dibentuk agar dapat mengolah data yang terstruktur dan tidak terstruktur kemudian menggunakan spark untuk melakukan implementasi analysis Machine Learning, berikut tahapan yang akan digunakan:

* 1. Melakukan *scraping* data twitter dengan menggunakan tweepy. Sebelumnya harus menyediakan access token dan API key untuk mengakses data Twitter tersebut.
  2. Hasil scraping kemudian di simpan dalam csv file. Data di csv file tersebut di input sebagai dataframe dalam format databrick. Data dibersihkan dengan proses preprocessing. Kemudian data diberi label sentimen dengan textblob.
  3. Kemudian data akan di input sebagai pyspark sql dataframe. Selanjutnya akan dilakukan klasifikasi analisis sentimen pada opini yang diberikan masyarakat terkait pandemi Covid. Logistic Regression dan N-Gram. Klasifikasi tersebut di evaluasi dan menghasilkan nilai akurasi.
  4. Memvisualisasikan data, Pada tahapan ini, hasil klasifikasi sentimen analisis pada opini terkait pandemi Covid akan diolah kembali untuk divisualisasikan berdasarkan hasil klasifikasi sentimen analisis untuk mempermudah dalam memahami hasil pengolahan data.

# **MACHINE LEARNING PIPELINE**

Pada bab ini akan dijelaskan Machine Learning Pipeline yang akan digunakan dalam sentiment analysis - Tweet opini pandemi Covid.



Gambar 2 Machine learning pipeline

Berikut tahapan Machine Learning Pipeline yang akan digunakan sebagai berikut:

1. Tokenizer

Tokenizer merupakan proses pemisahan teks menjadi kata, frasa, simbol atau elemen bermakna lainnya yang disebut dengan token. Tujuannya adalah mengeksplorasi kata kata dalam sebuah kalimat. Daftar token menjadi masukan untuk diproses lebih lanjut. Tokenisasi adalah proses untuk memotong dokumen menjadi pecahan kecil yang dapat berupa bab, sub-bab, paragraf, kalimat, dan kata (token). Pada proses ini akan menghilangkan whitespace .

1. Hashing TF

HashingTF merupakan proses untuk melakukan transformer yang nantinya akan mengambil kata dan mengubah kata tersebut menjadi suatu vektor dengan panjang yang tepat.

1. Logistic Regression Model

karena LogisticRegression adalah Estimator, Pipeline pertama-tama memanggil LogisticRegression.fit() untuk menghasilkan LogisticRegressionModel. Jika Pipeline memiliki lebih banyak Estimator, maka akan memanggil metode transform() LogisticRegressionModel pada DataFrame sebelum meneruskan DataFrame ke tahap berikutnya.

# **IMPLEMENTASI**

Pada bab ini akan dijelaskan mengenai lingkungan implementasi sentiment analysis - Covid (Klasifikasi).

1. Library yang digunakan

|  |
| --- |
| import pyspark  from pyspark.sql import DataFrame  from pyspark.ml.classification import LogisticRegression, LinearSVC, NaiveBayes  from nltk.tokenize import word\_tokenize  import tweepy  from tweepy import OAuthHandler  from tweepy import Stream  from collections import Counter  from plotly import graph\_objs as go  from wordcloud import WordCloud  import findspark  findspark.init()  import pyspark as ps  import warnings  from pyspark.sql import SQLContext |

Kode 1 Library yang digunakan

1. Start a Session

|  |
| --- |
| #Memulai Spark Session  try:  from pyspark.sql import SparkSession  sc = ps.SparkContext('local[2]')  sqlContext = SQLContext(sc)  spark = SparkSession.builder \  .master("local[2]") \  .appName("Proyek\_Sentiment Analysis") \  .getOrCreate()  except ValueError:  warnings.warn("SparkContext already exists in this scope") |

Kode 2 Start a session

1. Configuration

|  |
| --- |
| APIKey = "QnupptSEsqShuukhaPGsk0Svf"  APISecretKey = "HuD4bnPtEZcE1IVFDcluzuYcngbSMMYxKyfYA8MTXPUYZvn4N0"  AccessToken = "1390698546353999878-KExzyOWL2qm1zfmxZdrYmcnkaVbjue"  AccessTokenSecret = "dkEkoVa4QwW3Z8Rzvau6r37C7DhnUPU2zu91P6GZmm7Ec"  BearerToken ="AAAAAAAAAAAAAAAAAAAAALP5cAEAAAAACkdJeE7i23YH3L4qeYWpvoeqZAA%3DoOefkiPYcftGrIzdm9YgPi7hnYrI6ymuojs6031Dd5K5U7CWul" |

Kode 3 Configuration

1. Set Up

|  |
| --- |
| def connectOAuth():  auth = OAuthHandler(APIKey, APISecretKey)  auth.set\_access\_token(AccessToken, AccessTokenSecret)  api = tweepy.API(auth, wait\_on\_rate\_limit=True)  return api |

Kode 4 Set up

1. Scrapping

|  |
| --- |
| def scraptweets(search\_words, date\_since, date\_until):  db\_tweets = pd.DataFrame(columns=['username', 'tweetcreatedts', 'text'])  tweets = tweepy.Cursor(  API.search,  q=search\_words,  lang="id",  since=date\_since,  until=date\_until,  tweet\_mode='extended').items(5000)  tweet\_list = [tweet for tweet in tweets]  for tweet in tweet\_list:  username = tweet.user.screen\_name  tweetcreatedts = tweet.created\_at  try:  text = tweet.retweeted\_status.full\_text  except AttributeError:  text = tweet.full\_text  ith\_tweet = [username, tweetcreatedts, text]  db\_tweets.loc[len(db\_tweets)] = ith\_tweet  print('Proses Scrapping Selesai Dengan Jumlah Data', len(db\_tweets))  filename = 'reviews.csv'  db\_tweets.to\_csv(filename, index=False) |

Kode 5 Scrapping

1. Load Raw Data Twitter

|  |
| --- |
| df = pd.read\_csv('/content/reviews.csv', delimiter=",")  print(type(df))df.describe()  df.head(10)  df.isnull().sum() |

Kode 6 Load raw data twitter

1. Cleaning Tweets

|  |
| --- |
| def cleaningText(text):  text = re.sub(r'@[A-Za-z0-9]+', '', text) # remove mentions  text = re.sub(r'#[A-Za-z0-9]+', '', text) # remove hashtag  text = re.sub(r'RT[\s]', '', text) # remove RT  text = re.sub(r"http\S+", '', text) # remove link  text = re.sub(r'[0-9]+', '', text) # remove numbers  text = text.replace('\n', ' ') # replace new line into space  text = text.translate(str.maketrans('', '', string.punctuation)) # remove all punctuations  text = text.strip(' ') # remove characters space from both left and right text  return text  def casefoldingText(text): # Converting all the characters in a text into lower case  text = text.lower()  return text  def tokenizingText(text): # Tokenizing or splitting a string, text into a list of tokens  text = word\_tokenize(text)  return text  def filteringText(text): # Remove stopwors in a text  listStopwords = set(pd.read\_csv('/content/stopwords-id.txt'))  filtered = []  for txt in text:  if txt not in listStopwords:  filtered.append(txt)  text = filtered  return text  def toSentence(list\_words): # Convert list of words into sentence  sentence = ' '.join(word for word in list\_words)  return sentence  # Preprocessing tweets data  df['text\_clean'] = df['text'].apply(cleaningText)  df['text\_clean'] = df['text\_clean'].apply(casefoldingText)  df.drop(['text'], axis = 1, inplace = True)  df['text\_preprocessed'] = df['text\_clean'].apply(tokenizingText)  df['text\_preprocessed'] = df['text\_preprocessed'].apply(filteringText)  df['tweets'] = df['text\_preprocessed'].apply(toSentence)  # drop duplicates/spams tweets  df.drop\_duplicates(subset = 'tweets', inplace = True)  df.drop(['text\_clean', 'text\_preprocessed'], axis = 1, inplace = True) |

Kode 7 Cleaning tweets

1. Labeling Data

|  |
| --- |
| from textblob import TextBlob  def sentiment\_calc(tweets):  p = TextBlob(tweets).sentiment.polarity  if p<0 :  return("negatif")  elif p>0 :  return("positif")  else:  return("netral")  df['sentiment'] = df['tweets'].apply(sentiment\_calc) |

Kode 8 Labeling data

1. DataFrame Pyspark SQL

|  |
| --- |
| df = sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferschema='true').load('/content/preprocessed\_tweets.csv')  type(df) |

Kode 9 Dataframe Pyspark SQL

1. Split Train Test

|  |
| --- |
| (train\_set, val\_set, test\_set) = df.randomSplit([0.80, 0.10, 0.10], seed = 1103) |

Kode 10 Split train test

1. HashingTF + IDF + Logistic Regressio

|  |
| --- |
| from pyspark.ml.classification import LogisticRegression  lr = LogisticRegression(maxIter=100)  lrModel = lr.fit(train\_df)  predictions = lrModel.transform(val\_df)  from pyspark.ml.evaluation import MulticlassClassificationEvaluator  evaluator = MulticlassClassificationEvaluator()  evaluator.evaluate(predictions)  %%time  from pyspark.ml.feature import CountVectorizer  tokenizer = Tokenizer(inputCol="tweets", outputCol="words")  cv = CountVectorizer(vocabSize=262144, inputCol="words", outputCol='cv')  idf = IDF(inputCol='cv', outputCol="features", minDocFreq=5) #minDocFreq: remove sparse terms  label\_stringIdx = StringIndexer(inputCol = "sentiment", outputCol = "label")  lr = LogisticRegression(maxIter=100)  pipeline = Pipeline(stages=[tokenizer, cv, idf, label\_stringIdx, lr])  pipelineFit = pipeline.fit(train\_set)  predictions = pipelineFit.transform(val\_set)  Taccuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(val\_set.count())  Troc\_auc = evaluator.evaluate(predictions)  print ("Accuracy Score: {0:.4f}".format(Taccuracy))  print ("ROC-AUC: {0:.4f}".format(Troc\_auc)) |

Kode 11 HashingTF + IDF + Logistic Regression

1. CountVectorizer + IDF + Logistic Regression

|  |
| --- |
| %%time  from pyspark.ml.feature import CountVectorizer  tokenizer = Tokenizer(inputCol="tweets", outputCol="words")  cv = CountVectorizer(vocabSize=262144, inputCol="words", outputCol='cv')  idf = IDF(inputCol='cv', outputCol="features", minDocFreq=5) #minDocFreq: remove sparse terms  label\_stringIdx = StringIndexer(inputCol = "sentiment", outputCol = "label")  lr = LogisticRegression(maxIter=100)  pipeline = Pipeline(stages=[tokenizer, cv, idf, label\_stringIdx, lr])  pipelineFit = pipeline.fit(train\_set)  predictions = pipelineFit.transform(val\_set)  Caccuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(val\_set.count())  Croc\_auc = evaluator.evaluate(predictions)  print ("Accuracy Score: {0:.4f}".format(Caccuracy))  print ("ROC-AUC: {0:.4f}".format(Croc\_auc)) |

Kode 12 CountVectorizer + IDF + Logistic Regression

1. N Gram

|  |
| --- |
| from pyspark.ml.feature import NGram, VectorAssembler  from pyspark.ml.feature import ChiSqSelector  def build\_trigrams(inputCol=["tweets","sentiment"], n=3):  tokenizer = [Tokenizer(inputCol="tweets", outputCol="words")]  ngrams = [  NGram(n=i, inputCol="words", outputCol="{0}\_grams".format(i))  for i in range(1, n + 1)  ]  cv = [  CountVectorizer(vocabSize=262144,inputCol="{0}\_grams".format(i),  outputCol="{0}\_tf".format(i))  for i in range(1, n + 1)  ]  idf = [IDF(inputCol="{0}\_tf".format(i), outputCol="{0}\_tfidf".format(i), minDocFreq=5) for i in range(1, n + 1)]  assembler = [VectorAssembler(  inputCols=["{0}\_tfidf".format(i) for i in range(1, n + 1)],  outputCol="rawFeatures"  )]  label\_stringIdx = [StringIndexer(inputCol = "sentiment", outputCol = "label")]  selector = [ChiSqSelector(numTopFeatures=50,featuresCol='rawFeatures', outputCol="features")]  lr = [LogisticRegression(maxIter=100)]  return Pipeline(stages=tokenizer + ngrams + cv + idf+ assembler + label\_stringIdx+selector+lr)  %%time  trigram\_pipelineFit = build\_trigrams().fit(train\_set)  predictions = trigram\_pipelineFit.transform(val\_set)  Naccuracy = predictions.filter(predictions.label == predictions.prediction).count() / float(val\_set.count())  Nroc\_auc = evaluator.evaluate(predictions)  # print accuracy, roc\_auc  print ("Accuracy Score: {0:.4f}".format(Naccuracy))  print ("ROC-AUC: {0:.4f}".format(Nroc\_auc)) |

Kode 13 Ngram

1. N Gram Without Chi Squared Feature Selection

|  |
| --- |
| def build\_ngrams\_wocs(inputCol=["tweets","sentiment"], n=3):  tokenizer = [Tokenizer(inputCol="tweets", outputCol="words")]  ngrams = [  NGram(n=i, inputCol="words", outputCol="{0}\_grams".format(i))  for i in range(1, n + 1)  ]  cv = [  CountVectorizer(vocabSize=262144,inputCol="{0}\_grams".format(i),  outputCol="{0}\_tf".format(i))  for i in range(1, n + 1)  ]  idf = [IDF(inputCol="{0}\_tf".format(i), outputCol="{0}\_tfidf".format(i), minDocFreq=5) for i in range(1, n + 1)]  assembler = [VectorAssembler(  inputCols=["{0}\_tfidf".format(i) for i in range(1, n + 1)],  outputCol="features"  )]  label\_stringIdx = [StringIndexer(inputCol = "sentiment", outputCol = "label")]  lr = [LogisticRegression(maxIter=100)]  return Pipeline(stages=tokenizer + ngrams + cv + idf+ assembler + label\_stringIdx+lr)  %%time  trigramwocs\_pipelineFit = build\_ngrams\_wocs().fit(train\_set)  predictions\_wocs = trigramwocs\_pipelineFit.transform(val\_set)  accuracy\_wocs = predictions\_wocs.filter(predictions\_wocs.label == predictions\_wocs.prediction).count() / float(val\_set.count())  roc\_auc\_wocs = evaluator.evaluate(predictions\_wocs)  # print accuracy, roc\_auc  print ("Accuracy Score: {0:.4f}".format(accuracy\_wocs))  print ("ROC-AUC: {0:.4f}".format(roc\_auc\_wocs))  test\_predictions = trigramwocs\_pipelineFit.transform(test\_set)  test\_accuracy = test\_predictions.filter(test\_predictions.label == test\_predictions.prediction).count() / float(test\_set.count())  test\_roc\_auc = evaluator.evaluate(test\_predictions)  # print accuracy, roc\_auc  print ("Accuracy Score: {0:.4f}".format(test\_accuracy))  print ("ROC-AUC: {0:.4f}".format(test\_roc\_auc)) |

Kode 14 N Gram Without Chi Squared Feature Selection

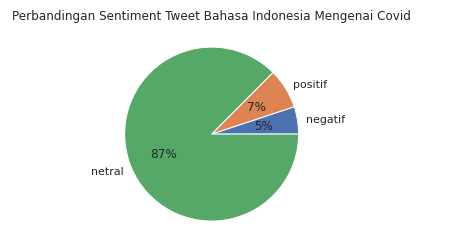
1. Visualisasi Data

|  |
| --- |
| import pyspark.sql.functions as func  sentiment = df.groupBy('sentiment').count().select(func.col("sentiment").alias("distinct\_sentiment"),func.col("count").alias("country\_sentiment"))  sentiment.show()  import matplotlib.pyplot as plt  senti = sentiment.toPandas()  fig = plt.pie(senti['country\_sentiment'], labels=senti['distinct\_sentiment'])  plt.title('Perbandingan Sentiment Tweet Bahasa Indonesia Mengenai Covid')  plt.show() |

Kode 15 Visualisasi Data

# **HASIL**

Hasil dari penelitian ini dapat dilihat dari gambar dibawah ini:



Perbandingan Sentiment Tweet bahasa indonesia mengenai covid yakni:

|  |  |  |
| --- | --- | --- |
| Netral | Positif | Negatif |
| 87% | 7% | 5% |

# **KESIMPULAN**

Kesimpulan :

Data tweets diklasifikasikan kedalam tiga label, yakni positif, negatif, dan netral. Model klasifikasi terbaik dari implementasi adalah N-Gram dengan nilai akurasi sebesar 0.

Sentimen masyarakat yang diwakili data twitter berbahasa indonesia mengenai ‘Covid’ adalah netral, hal ini dibuktikan dengan nilai persentase dari label netral yakni sebesar 87%

Saran :

Analisis masih memiliki ruang pengembangan dengan penggunaan model klasifikasi lainnya yang merupakan state-of-the-art untuk klasifikasi. Untuk mengolah data bahasa indonesia, data perlu diolah dengan transformasi tertentu, untuk dapat melakukan klasifikasi.

# **REFRENSI**

|  |  |
| --- | --- |
| [1] | "Virtual press conference on COVID-19 – 11 March 2020," 2020. |
| [2] | "we are social," 31 Januari 2019. [Online]. Available: https://wearesocial.com/uk/blog/2019/01/digital-in-2019-global-internet-use-accelerates/. [Accessed 26 Mei 2022]. |
| [3] | S. Pramana, B. Yuniarto, S. Mariyah, I. Santoso and R. Nooraeni, Data Mining dengan R : Konsep Serta Implementasi, Bogor: In Media, 2018. |
| [4] | G. A. Buntoro, "Analisis Sentimen Calon Gubernur DKI Jakarta 2017 Di Twitter," *Integer Journal,* vol. 2, no. 1, p. 2, 2017. |
| [5] | R. Madan, "Medium," 31 Mei 2019. [Online]. Available: https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3. [Accessed 17 Mei 2022]. |