Sentiment Analysis with Logistic Regression

Oleh:

2301878624 - Vincent Ciptadi
2301893985 - Gabrielle Angelica I
2301901343 - Rhenald Ariendra P

Link video presentasi:
https://discord.com/channels/@me/840830589950230528/860896585479094272

Latar Belakang

Media sosial meupakan sebuah wadah yang memfasilitasi dalam pertukaran informasi maupun penyampaian opini dengan waktu yang cepat serta cakupan wilayah yang luas. Penggunaan media sosial sangat populer dikalangan masyarakat, hal ini sudah menjadi hal yang biasa. Terlebih lagi, pada era pandemik saat ini semakin banyak orang yang menggunakan media sosial, maka dari itu semakin banyak informasi yang beredar di media sosial, semakin banyak pula opini dari netizen tentang informasi atau berita yang beredar. Maka dari itu, diperlukan pengawasan terhadap penggunaan media sosial untuk memahami dan juga mengelompokkan sifat (positif, negatif, dan netral) yang terdapat pada cuitan netizen dalam bermedia sosial.

Pada kesempatan kali ini, media sosial yang akan kami bahas yaitu Twitter. Twitter merupakan layanan media sosial dan *microblog online* yang memungkinkan setiap penggunanya untuk mengirim dan membaca setiap pesan dalam bentuk teks hingga 280 karakter dan dikenal dengan sebutan *tweet*. Twitter sendiri menjadi sebuah sarana yang dimanfaatkan untuk berbagai macam keperluan, seperti sarana protes, kampanye, sarana pembelajaran, dan juga media komunikasi. Sebagian besar orang mengekspresikan perasaan mereka terhadap konteks saat ini.

Sebagai manusia, kita tentu bisa menebak sebuah *sentiment* apakah itu positif atau juga negatif. Namun, bagaimana jika terdapat data yang sangat banyak? Akan lebih baik jika menggunakan teknologi yang ada salah satunya *sentiment analysis*. *Sentiment analysis* merupakan sebuah studi yang bertujuan untuk mengolah data tekstual dan mendapatkan informasi yang terkandung dalam kalimat teks tersebut. Pada umumnya, ketika kita melakukan *sentiment analysis*, maka kalimat teks dari data yang ada nantinya akan dikelompokkan lalu selanjutnya dikemukakan apakah kalimat teks itu bersifat positif, negatif, ataupun netral. *Sentiment analysis* sendiri dikelompokkan menjadi dua jenis, yaitu berdasarkan fakta (objektif) dan berdasarkan pendapat (subjektif). *Sentiment analysis* yang berdasarkan fakta merupakan sebuah pernyataan objektif terkait suatu entitas ataupun atributnya. Salah satu contohnya, yaitu “Saya membeli mobil itu kemarin”. Sedangkan *sentiment analysis* yang berdasarkan pendapat merupakan sebuah pernyataan mengenai *sentiment*, emosi, pendapat atau perasaan terhadap suatu entitas ataupun atributnya.

Terdapat berbagai macam algoritma yang dapat digunakan untuk melakukan *sentiment analysis*. Beberapa diantaranya yaitu *k-Nearest Neighbor* (k-NN), *Naïve Bayes Classifier* (NBC), *Logistic Regression*, dan lainnya. K-NN adalah sebuah algoritma yang bergantung pada jarak untuk melakukan klasifikasi, jika fitur memiliki skala yang berbeda, maka untuk menormalisasikan data pelatihan dapat meningkatkan keakuratannya secara dramatis. NBC menggunakan perhitungan probabilitas untuk melakukan klasifikasi. *Logistic Regression* dapat mengukur hubungan antara variabel *target* yang ingin diprediksi dan juga variabel *input* atau fitur yang akan digunakan dengan fungsi logistik, perhitungan probabilitas akan dihitung menggunakan fungsi sigmoid untuk mengubah nilai menjadi 0 atau 1.

Tujuan dan Manfaat

Tujuan dan manfaat dari *sentiment analysis* ini yaitu:

1. Melakukan *sentiment analysis* dengan menggunakan *Logistic Regression*.
2. Mengelompokkan kalimat menjadi beberapa sifat seperti positif, negatif, dan netral.
3. Mengetahui *trend* apa saja yang sedang terjadi pada dunia nyata.

Data

Dataset untuk studi ini didapatkan melalui Natural Language Toolkit (NLTK) yang merupakan sebuah library pada Python. NLTK merupakan sebuah kumpulan sumber daya untuk Python yang dapat digunakan untuk pemrosesan teks, klasifikasi, *tagging*, dan tokenisasi. Toolbox ini memainkan peran kunci dalam mengubah data teks dalam tweet ke dalam format yang dapat digunakan untuk mengekstrak *sentiment* dari tweet tersebut. NLTK menyediakan berbagai macam fungsi yang dapat digunakan saat pre-processing data sehingga data yang didapat dari twitter bisa disesuaikan.

Disini kami menggunakan korpus “twitter_sampels” dari NLTK sebagai dataset. Korpus twitter_samples telah mengkategorikan sifat dari tweet-tweet yang ada. Dataset ini mengandung 5000 tweet yang bersifat positif, 5000 tweet bersifat negatif, dan 20000 tweet yang bersifat netral atau dengan kata lain tidak terdeteksi positif atau negatifnya.

Metodologi Penelitian

Pada umumnya, kebanyakan orang menggunakan metode *Naïve Bayes* untuk melakukan studi ini. Namun, metode yang kami gunakan yaitu *Logistic Regression*. Menurut sebuah penelitian dari *University of California*, Hal pembeda utama dari *Naïve Bayes* dan *Logistic Regression* yaitu jika dataset semakin besar, maka *naïve bayes* akan menjadi kurang relevan. Sedangkan jika ukuran dataset masih kecil, maka *naïve bayes* akan lebih cepat mencapai solusinya. Lalu, *naïve bayes* juga akan mengasumsikan semua variabel yang ada yaitu *independent*, meski pada kenyataannya tidak ada data yang *independent*, hanya bisa mendekati *independent*. Hal itulah yang menyebabkan *logistic regression* menjadi lebih baik dibandingkan *naïve bayes*.

Pada umumnya, jika dapat menggunakan metode *logistic regression* dengan benar, hasil akurasi yang dicapai akan menyentuh kurang lebih sekitar 90%.

Berikut merupakan tahapan-tahapan yang dilakukan dalam penelitian ini:

1. **Pengumpulan data**
   Dilakukan dengan menggunakan korpus “twitter_samples” dari library NLTK.

2. **Split data**
   Pembagian dataset yaitu sebesar 80% untuk data *training* dan 20% untuk data *testing*.

3. **Pre-processing**
   Menghilangkan *stopwords*, *punctuation*, dan kata-kata seperti RT/ hyperlinks/ hashtag
   Contoh:
   Sebelum pre-processing:
   #FollowFriday @wncer1 @Defense_gouv for being top influencers in my community this week :)

   Setelah pre-processing:
   ['followfriday', 'top', 'influenc', 'commun', 'week', ':)']

4. **Analisis dan validasi**
   Melakukan feature extraction dan menggunakan logistic regression dengan activation function sigmoid dan juga optimizer gradient descent.

Hasil Eksperimen dan Analisis

Pengkodean untuk eksperimen ini dilakukan menggunakan bantuan Google Colaboratory. Berikut ini merupakan hasilnya:

Pertama, kita akan mengimport semua library yang akan digunakan dan print salah satu contoh dari dataset

```
[1]  import nltk
     import numpy as np
     import pandas as pd
     from nltk.corpus import twitter_samples
     nltk.download('twitter_samples')
```

```
[2]  token = twitter_samples.tokenized('positive_tweets.json')
     print(token[0])
```

```
['#FollowFriday', '@France_Inte', '@PKuchly57', '@Milipol_Paris', 'for', 'being', '
```

Setelah itu kita akan mengunduh komponen dari library NLTK dan melakukan POS Tagging

```
[3]  nltk.download('wordnet')
     nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

Menentukan POS Tagging dari dataset tersebut

```
[4]  from nltk.tag import pos_tag
     print(pos_tag(token[0]))
```

```
[('#FollowFriday', 'JJ'), ('@France_Inte', 'NNP'), ('@PKuchly57', 'NNP'), ('@Milipc
```

Selanjutnya, dataset dibagi menjadi menjadi data training dan data testing. Data training sebesar 80% dan data testing sebesar 20%.

```
[5]  all_positive_tweet = twitter_samples.strings('positive_tweets.json')
     all_negative_tweet = twitter_samples.strings('negative_tweets.json')

     positive_train = all_positive_tweet[:4000]
     positive_test = all_positive_tweet[4000:]
     negative_train = all_negative_tweet[:4000]
     negative_test = all_negative_tweet[4000:]

     x_train = positive_train + negative_train
     x_test = positive_test + negative_test

     y_train = np.concatenate((np.ones(len(positive_train)), np.zeros(len(negative_tra
     y_test = np.concatenate((np.ones(len(positive_test)), np.zeros(len(negative_test)
```

```
[8]  print(positive_train[50])
     print(negative_train[30])

     @groovinshawn they are rechargeable and it normally comes with a charger when u buy
     @bbygjrlmgc oh :( i hate when that happens i get so sad over it too
```

```
[9]  print(len(x_train))
     print(len(x_test))

     8000
     2000
```

Langkah selanjutnya yaitu pre-processing data. Kita akan menghilangkan noise dari dataset menggunakan regex, melakukan stemming data, dan melakukan tokenize pada tweetnya. Lalu membuat dictionary untuk memproses lebih lanjut data training tersebut.

```
[11] import re, string
     from nltk.corpus import stopwords
     from nltk.stem import PorterStemmer
     from nltk.tokenize import TweetTokenizer

     def processing(tweets):
       stem = PorterStemmer()
       stopword = stopwords.words('english')

       tweets = re.sub(r'^RT[\s]+', '', tweets)
       tweets = re.sub(r'https?:\/\/.*[\r\n]*', '', tweets)
       tweets = re.sub(r'#', '', tweets)

       tokenizer = TweetTokenizer(preserve_case=False, reduce_len=True, strip_handles=T
       tokenize_tweet = tokenizer.tokenize(tweets)

       cleaned_tweet = []
       for tokenize in tokenize_tweet:
         if tokenize not in stopword and tokenize not in string.punctuation:
           stemmed_token = stem.stem(tokenize)
           cleaned_tweet.append(stemmed_token)

       return cleaned_tweet
```

```
[12] def build_freqs(tweets, ys):
       # convert array to list
       ys_list = np.squeeze(ys).tolist()

       freqs = {}
       for y, tokenize in zip(ys_list, tweets):
         for tokens in processing(tokenize):
           double = (tokens, y)
           if double in freqs:
             freqs[double] += 1
           else:
             freqs[double] = 1

       return freqs
```

```
[13] freqs = build_freqs(x_train, y_train)
```

Lalu, mengaktifkan activation function dimana kita menggunakan sigmoid dan juga optimizer yatu gradient descent.

```
[15] def sigmoid(tweet):
       return 1/(1 + np.exp(-tweet))
```

```
[16] def GradientDescent(tweet_x, tweet_y, teta, alpha, epoch, batch_epoch, batch_size):
       temp = tweet_x.shape[0]
       error = []

       for i in range(1, epoch+1):
         for j in range(batch_epoch):
           batch = np.random.randint(0, temp, size=batch_size)
           x_train = tweet_x[batch, :]
           y_train = tweet_y[batch, :]

           z = np.dot(x_train, teta)
           activation = sigmoid(z)

           cost_function = -1/batch_size * (np.dot(np.transpose(y_train), np.log(activation)) + np.dot(np.transpose(1 - y_train), np.log(1 - activation)))

           teta = teta - (alpha/temp * np.dot(np.transpose(x_train), (activation - y_train)))

         if not i % 10 and i:
           print(f'Epoch: {i}, Loss: {np.squeeze(cost_function)}')

         error.append(cost_function)

       cost_function = float(cost_function)
       return cost_function, teta, error
```

Selanjutnya kita akan melakukan feature extraction yaitu mengambil sesuatu yang unik dari data yang akan diolah nantinya.

```
[17] def feature_extraction(tweets, freqs):
       cleaned = processing(tweets)

       # Create Vector 1x3
       vector = np.zeros((1,3))
       # bias
       vector[0,0] = 1

       for tokenize in cleaned:
         positive_double = (tokenize, 1.0)
         negative_double = (tokenize, 0.0)

         # increment for positive label 1 and negative lable 0
         if positive_double in freqs.keys():
           vector[0,1] += freqs[positive_double]
         if negative_double in freqs.keys():
           vector[0,2] += freqs[negative_double]

       assert(vector.shape == (1,3))
       return vector
```

```
[18] def predict(tweets, freqs, teta):
       feature = feature_extraction(tweets, freqs)
       y_pred = sigmoid(np.dot(feature, teta))

       return y_pred
```

Kita juga akan membangun model logistic regression yang akan digunakan

```
[19] def logistic_regression(x_tweet, y_tweet, freqs, teta):
       y_hat = []
       temp = y_tweet.shape[0]

       for tokens in x_tweet:
         y_pred = predict(tokens, freqs, teta)

         if y_pred > 0.5:
           y_hat.append(1)
         else:
           y_hat.append(0)

       y_hat = np.array(y_hat)
       y_hat = np.reshape(y_hat, (temp, 1))

       acc = np.sum(y_hat == y_tweet) / temp

       return acc
```

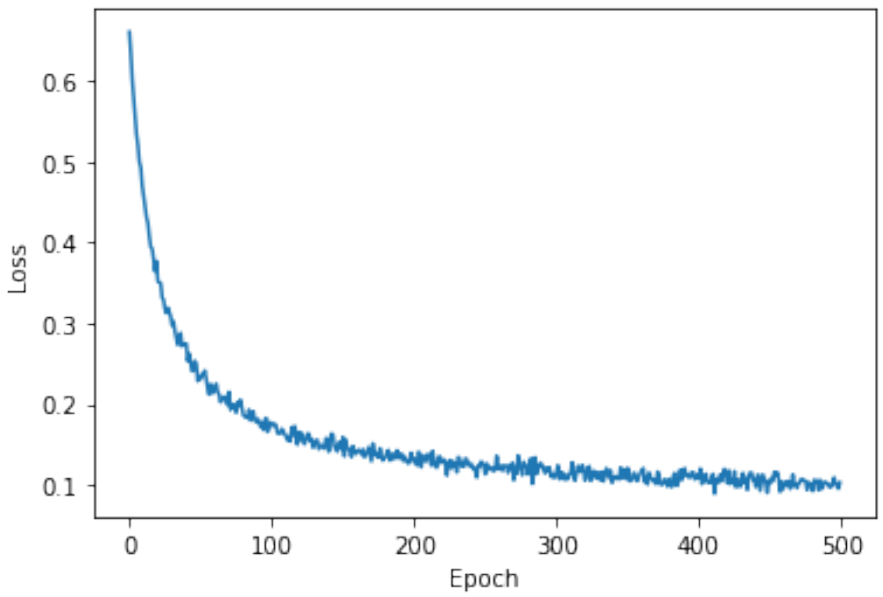
Lalu dari data yang ada, kita bagi menjadi 2 yaitu untuk features dan juga target. Dan juga mengacaknya agar data tidak terpilih berurutan.

```
[20] features = np.zeros((len(x_train), 3))
     for i in range(len(x_train)):
       features[i, :] = feature_extraction(x_train[i], freqs)

     target = y_train
     target = np.reshape(target, (-1,1))
```

```
[21] random_feature = np.random.permutation(range(len(features)))
     features = features[random_feature]
     target = target[random_feature]
```

Melakukan print epoch dan loss dengan optimizer Gradient Descent dengan epoch 500 dan menghasilkan cost after training 0.1031

```
[22] cost_function, teta, error = GradientDescent(features, target, teta=np.zeros((3,1
     print(f'Cost after Training: {cost_function:.4f}')

     Epoch: 10, Loss: 0.4705514782081134
     Epoch: 20, Loss: 0.37673079204276416
     Epoch: 30, Loss: 0.30857017660090796
     Epoch: 40, Loss: 0.2742001959562107
     Epoch: 50, Loss: 0.2347718830463905
```



Setelah selesai, kita dapat menampilkan hasil akurasi dari eksperimen yang telah dilakukan. Akurasi yang kami dapatkan yaitu sebesar 99,45%

```
[23] total_accuracy = logistic_regression(x_test, np.reshape(y_test, (-1,1)), freqs, te
     print(f'Accuracy: {total_accuracy * 100}%')

     Accuracy: 99.45%
```

Simpulan

Sentiment analysis bertujuan untuk mengolah data tekstual dan mendapatkan informasi yang terkandung dalam kalimat teks tersebut. Teks akan dikelompokkan lalu dikemukakan apakah kalimat teks itu bersifat positif, negatif, ataupun netral. Nilai akurasi dari penggunaan metode logistic regression untuk sentiment analysis ini yaitu sebesar 99,45%.

Referensi

1. Sinha, A. K. (2020, August 1). *Twitter Sentiment Analysis (NLTK+logistic reg)*. Kaggle. https://www.kaggle.com/ashishsinha5/twitter-sentiment-analysis-nltk-logistic-reg.
2. Daityari, S. (2021, January 28). *How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)*. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk.
3. Trojak, J. (2020, December 7). *Sentiment analysis with logistic regression in Python with nltk library*. Medium. https://joannatrojak.medium.com/sentiment-analysis-with-logistic-regression-in-python-with-nltk-library-d5030b1d84e3.
4. Ng, A., & Jordan, M.I. (2001). On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. *NIPS*.
5. Afham, M. (2021, June 17). *Twitter Sentiment Analysis using NLTK, Python - Towards Data Science*. Medium. https://towardsdatascience.com/twitter-sentiment-analysis-classification-using-nltk-python-fa912578614c.
6. BINUS UNIVERSITY. (2020, June 12). *Pengenalan Sentiment Analysis*. BINUS UNIVERSITY MALANG | Pilihan Universitas Terbaik Di Malang. https://binus.ac.id/malang/2020/06/pengenalan-sentiment-analysis/.