LAMPIRAN

Lampiran 1, Proses Cleaning dan Case Folding pada Python

```
def clean_text(text):
    # Hapus username
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    # hapus hashtag
    text = re.sub(r'#[A-Za-z0-9]+', '', text)
    # Hapus URL
    text = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-
_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', text)
    # Hapus tanda baca
    text = re.sub(r"[^a-zA-Z]", " ", text)
    # Hapus angka
    text = re.sub(r"\d+", "", text)
    # Hapus spasi ekstra
    text = re.sub('\s+', ' ', text).strip()
    # Lowercase
    text = text.lower()
    # # Normalisasi slang words
    # text = ' '.join(slang_words_dict.get(word, word) for
word in text.lower().split())
    # Hapus emoji
    emoj = re.compile("["
                      u"\U0001F600-\U0001F64F"  # emoticon
                      u"\U0001F300-\U0001F5FF"  # simbol &
piktograf
                      u"\U0001F680-\U0001F6FF"  # transport &
simbol peta
                      u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                      u"\U00002500-\U00002BEF"  # karakter
china
                      u"\U00002702-\U000027B0"
                      u"\U00002702-\U000027B0"
                      u"\U000024C2-\U0001F251"
                      u"\U0001f926-\U0001f937"
                      u"\U00010000-\U0010ffff"
                      u"\u2640-\u2642"
                      u"\u2600-\u2B55"
                      u"\u200d"
                      u"\u23cf"
                      u"\u23e9"
                      u"\u231a"
                      u"\ufe0f"  # dingbats
                      u"\u3030"
                      "]+", re.UNICODE)
```

```
    text = re.sub(emoj, '', text)
    text = re.sub(r'\bw+k+\b', '', text)
    return text

df['clean'] = [clean_text(i) for i in tweet]
tweet = df['clean']
df.head(10)
```

Lampiran 2. Proses *Tokenizing*

```
import nltk

nltk.download('punkt')

nltk.download('punkt_tab')

tokens = tweet.apply(word_tokenize)

df['Token'] = df['clean'].apply(word_tokenize)

df.head(10)
```

Lampiran 3. Proses *Stopword Removal*

```
stop_words = set(stopwords.words("indonesian"))
additional_stop_words = set(['yuhuuuuuu', 'hooh', 'iyaaaaa',
'kuuuu', 'xixixi', 'nahhh', 'sipsipp', 'sksk', 'hehehehehe',
'jir', 'hadeuh', 'lah', 'buseeeee','hadeh', 'yaelahhhhh',
                        'cie', 'nya', 'nih', 'sih', 'si',
'tuh', 'ya', 'anjrit', 'huhu', 'youuu', 'wkwwkwk', 'coy',
'duh', 'okay', 'wkakakak', 'aduh', 'ssstt', 'wkwkwkwkwkw',
                        'hehe', 'wiiiw', 'lagiiiii', 'nan',
'loh', 'ckckck', 'ajg', 'ehhemm', 'hei', 'oiya', 'wkwkk',
'atuh', 'siii', 'anjay', 'wkwkwkwwk', 'wkwkwkkw', 'wow',
                        'wkwk','hahaha', 'wkkwkkwkkwkk', 'eh',
'deh', 'hah', 'anjrittt', 'woaaah', 'doang', 'assoii',
'yeayyy', 'woiiii', 'hhh', 'aswww', 'brou', 'hmmm', 'yak',
                        'woah', 'wkwkwk', 'lololol', 'wgwgwg',
'anjir', 'anjer', 'anjeng','zsazsa', 'woyyyyyy', 'wkwkwkw',
'wkwkw', 'oalah', 'aaaa', 'wkwkwkwkwkwk', 'kack',
```

```
                        'jderr', 'ayolaaah','wiiih',
'omonaaaaa', 'omo', 'omoo', 'ooo', 'aaaaaaaa', 'wii', 'wahhh',
'waeee', 'nyaaa', 'deg', 'ji', 'bbies', 'gelaseeeehhh',
                        'bi', 'yakaan', 'aduhh', 'eehh', 'adh',
'kaaaaan', 'hufttt', 'wey', 'coyyy', 'yakan','mih',
'blablabla', 'ush', 'lahhhh', 'njuk','brow', 'asdfgshshsjk',
                        'idih', 'sooo', 'siieeee', 'eeeh',
'gais', 'uaah', 'lyke', 'bambankkkk', 'kannn', 'duhh',
'busyettt', 'yaaaa', 'loch', 'hiks', 'broo', 'yeay',
'buseeeeet',
                        'heleh','sksksk','jirr', 'eww',
'hikz', 'hahahahahahaha', 'siiii', 'siy', 'ciahh', 'cuz',
'wakwaw', 'laah', 'hshs', 'omooo', 'heheh', 'wkowkwk',
'hahahha',
                        'wih', 'nich', 'awowkok', 'whoaaa',
'wowww', 'yakkkk','mwah', 'wkwkwkwkwk', 'yeah', 'woy',
'chuaaakkksss', 'ohhh', 'laaah', 'wkwkwkwkwkwkwk', 'kannnnn',
                        'lahh', 'ehhhh', 'yaaaaaa', 'ooh',
'gaiss','njinkk', 'wkwkwkwkw', 'yaampunnn', 'cieee',
'aaaaaak', 'awokawok','ahhhh', 'xixi', 'iya', 'lho', 'anj',
'hoh',
                        'nihh', 'anjeer', 'ebuseeet', 'brou',
'iiiiihhhh', 'hhii', 'ndul', 'wkwkwkwkwkwwkk', 'yeu', 'arghh',
'beb', 'assuuu', 'beuhhh', 'hoalah', 'njirr', 'hemmm',
                        'njiiir', 'siss', 'wakakakakak',
'yuuuhuuu', 'widihh', 'aaakk', 'aaaaah', 'waah', 'aih', 'akh',
'halahh', 'oo', 'tsay', 'hue', 'mwehehe', 'idihh', 'wkwwk',
                        'wehehe', 'atulah', 'se', 'cieee',
'oy' 'adh', 'sii', 'anjay', 'ebuset', 'assoii', 'nge', 'waw',
'sjjshsjkw', 'hihiihi', 'weh', 'blablabla', 'fa', 'fi', 'fu',
                        'hue', 'beuh', 'blabla', 'yuhu',
'blablabla', 'fafifu', 'wasweswos', 'hah', 'heh', 'hoh',
'huhu', 'woy', 'njr', 'kah', 'haha', 'wele', 'an', 'oy', 'oh',
                        'woi', 'mu', 'a', 'ber', 'in', 'ups',
'sc', 'bro', 'hi', 'yaelah', 'ah', 'ih', 'oalah', 'yekan',
'e', 'bla', 'halah', 'ok', 'hayo', 'shit', 'dih', 'yok',
                        'hmm', 'hayoloh', 'ye', 'hadah', 'uh',
'hey', 'y', 'c', 'yekan', 'yas', 'halah', 'walah', 'hi',
'lalala', 'euy', 'hey', 'epep', 'ku', 'tit', 'huft', 'ih',
                        'halah', 'hi', 'yups', 'elah', 'woi',
'brou', 'yak', 'nya', 'haduh', 'akh', 'sihh', 'ha', 'waao',
'blablablabla', 'ahh', 'yes', 'wihh', 'tuu', 'ew',
                        'yuk', 'widih', 'yakali', 'itu',
'dalam'])
stop_words.update(additional_stop_words)
# def stopword(text):
#     text = [word for word in text if word not in stop_words]
```

```
#     return text
def stopword(text):
    # Ubah ke string jika text bukan string
    if not isinstance(text, str):
        text = ' '.join(text)

    # Memisahkan teks menjadi kata-kata
    words = text.split()

    # Menghapus stop words
    filtered_words = [word for word in words if word not in
stop_words]

    return ' '.join(filtered_words)
df['Stopword'] = df['Normalisasi'].apply(stopword)
df.head(10)
```

Lampiran 4. Proses *Stemming*

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import swifter
# Daftar kata khusus yang ingin dipertahankan
kata_khusus = ["semoga", "setuju", "semua", "bali", "alasan",
"nangis", "agensi", "senang", "relasi", "nilai", "korelasi",
"busana", "ketik", "kesan", "pesan",
               "muda", "laris", "capai", "wisata",
"kunjungan"]

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# stemmed
def stemmed_wrapper(term):
    # Periksa apakah term ada dalam daftar kata khusus
    if term in kata_khusus:
        return term  # Jika term ada dalam daftar kata khusus,
kembalikan term tanpa melakukan stemming
    else:
        return stemmer.stem(term)
# Create a dictionary to store stemmed terms
term_dict = {}

# Iterate over each document in the DataFrame
for document in df['Stopword']:
```

```
    words = document.split()  # Split document into words
    for term in words:
        if term not in term_dict:
            term_dict[term] = ' '

# Apply stemming to each term and store in dictionary
for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)

# Print results
for term, stemmed in term_dict.items():
    print(f"{term} : {stemmed}")

print(f"Total unique terms: {len(term_dict)}")
print("--------------------")
```

Lampiran 5. Proses Pembuatan *Dictionary*

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import swifter
# Daftar kata khusus yang ingin dipertahankan
kata_khusus = ["semoga", "setuju", "semua", "bali", "alasan",
"nangis", "agensi", "senang", "relasi", "nilai", "korelasi",
"busana", "ketik", "kesan", "pesan",
               "muda", "laris", "capai", "wisata",
"kunjungan"]

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# stemmed
def stemmed_wrapper(term):
    # Periksa apakah term ada dalam daftar kata khusus
    if term in kata_khusus:
        return term  # Jika term ada dalam daftar kata khusus,
kembalikan term tanpa melakukan stemming
    else:
        return stemmer.stem(term)
# Create a dictionary to store stemmed terms
term_dict = {}

# Iterate over each document in the DataFrame
for document in df['Stopword']:
    words = document.split()  # Split document into words
    for term in words:
```

```
        if term not in term_dict:
            term_dict[term] = ' '

# Apply stemming to each term and store in dictionary
for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)

# Print results
for term, stemmed in term_dict.items():
    print(f"{term} : {stemmed}")

print(f"Total unique terms: {len(term_dict)}")
print("--------------------")
```

Lampiran 6. Proses TF-IDF

```
import joblib
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

# Inisialisasi TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=200,
                                   stop_words=None,
                                   lowercase=False,
                                   ngram_range=(1, 3))

# Melakukan pembobotan kata dengan TF-IDF pada data pelatihan
X_tfidf = tfidf_vectorizer.fit_transform(X)

# Menyimpan objek TfidfVectorizer
joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')

# Mendapatkan fitur (kata-kata) yang sesuai dengan indeksnya
dari TF-IDF
feature_names = tfidf_vectorizer.get_feature_names_out()

# Tampilkan contoh beberapa kata yang terbentuk
print("pariwisata, bali:", feature_names[:10])

# Membuat DataFrame dari matriks TF-IDF
tfidf_df = pd.DataFrame(X_tfidf.toarray(),
columns=feature_names)

# Menyimpan matriks TF-IDF sebagai file CSV
tfidf_df.to_csv('tfidf_matrix.csv', index=False)
```

```
# Menampilkan contoh matriks TF-IDF
print(tfidf_df.head())
```

Lampiran 7. Proses Pembagian Data Menggunakan *KFold*

```
# from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import StratifiedKFold

# # Inisialisasi TfidfVectorizer
# tfidf_vectorizer = TfidfVectorizer(max_features=200,
stop_words=None, lowercase=False, ngram_range=(1, 3))

# # Melakukan pembobotan kata dengan TF-IDF pada data
# X_tfidf = tfidf_vectorizer.fit_transform(X)

# Inisialisasi StratifiedKFold
skf = StratifiedKFold(n_splits=5, shuffle=True,
random_state=42)

for train_index, test_index in skf.split(X_tfidf, y):
    X_train, X_test = X_tfidf[train_index],
X_tfidf[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Lampiran 8. Proses Pemodelan Klasifikasi Sentimen

```
import numpy as np

# instantiate classifier with linear kernel
linear_svc=SVC(kernel='linear', probability=True)

# fit classifier to training set
linear_svc.fit(X_train,y_train)

# make predictions on test set
y_pred_test=linear_svc.predict(X_test)
# # compute and print accuracy score
print('Training set score:
{:.4f}'.format(linear_svc.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(linear_svc.score(X_test,
y_test)))
```

Lampiran 9. Proses Evaluasi Model Klasifikasi Sentimen

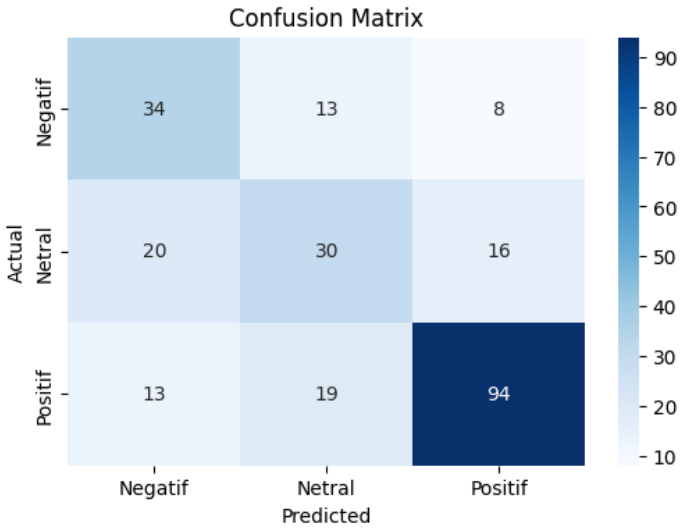
```
from sklearn.metrics import confusion_matrix,
classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Hitung dan tampilkan confusion matrix
conf_mat = confusion_matrix(y_test, y_pred_100)
print('\nConfusion Matrix:')
print(conf_mat)

# Tampilkan classification report
target_names = ['Negatif', 'Netral', 'Positif']
print('\nClassification Report:')
print(classification_report(y_test, y_pred_100,
target_names=target_names))

# Tampilkan confusion matrix menggunakan heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues',
xticklabels=['Negatif', 'Netral', 'Positif'],
yticklabels=['Negatif', 'Netral', 'Positif'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Lampiran 10. Pengujian Menggunakan Data Baru

```
import pandas as pd
import joblib
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt

# Memuat model SVM
model = joblib.load('model_svm.pkl')

# Memuat TfidfVectorizer
tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')

# Mengimpor dan menyiapkan data baru
```

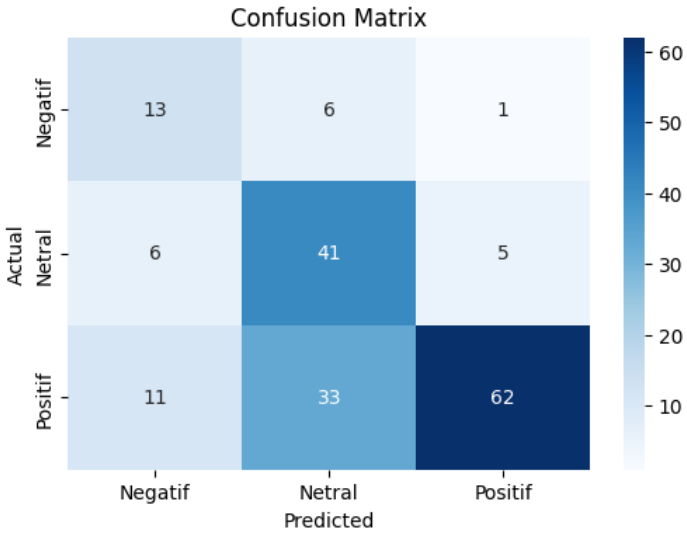
```
new_data = '/content/drive/My Drive/Colab
Notebooks/data/tweet_g20.csv'
df = pd.read_csv(new_data)

# Misalkan kolom yang berisi teks bernama 'Text' dan kolom
label bernama 'Label'
X_new_texts = df['Tweet']
y_true = df['Kelas'].map({'positif': 2, 'negatif': 0,
'netral': 1}).values
# Transformasi data teks baru menjadi fitur menggunakan TF-IDF
X_new_tfidf = tfidf_vectorizer.transform(X_new_texts)
# print(f'Tipe data y_true: {type(y_true.iloc[0])}, Tipe data
predictions: {type(predictions[0])}')

# Melakukan prediksi
predictions = model.predict(X_new_tfidf)
print(predictions)

# Menghitung akurasi
accuracy = accuracy_score(y_true, predictions)
print(f'Akurasi: {accuracy * 100:.2f}%')

# Membuat confusion matrix
conf_matrix = confusion_matrix(y_true, predictions)
print('Confusion Matrix:')
print(conf_matrix)

# Menampilkan classification report
class_report = classification_report(y_true, predictions)
print('Classification Report:')
print(class_report)

# Tampilkan confusion matrix menggunakan heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['Negatif', 'Netral', 'Positif'],
yticklabels=['Negatif', 'Netral', 'Positif'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Lampiran 11. Hasil Ealuasi Pengujian Model



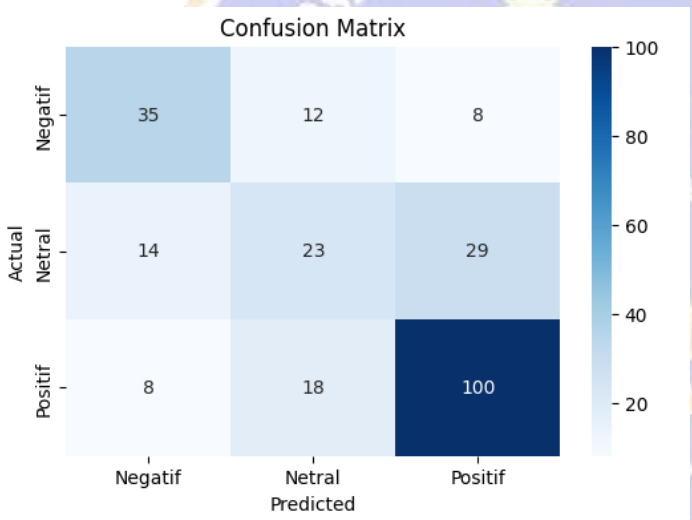
Classification Report:	precision	recall	f1-score	support
Negatif	0.51	0.62	0.56	55
Netral	0.48	0.45	0.47	66
Positif	0.80	0.75	0.77	126
accuracy			0.64	247
macro avg	0.60	0.61	0.60	247
weighted avg	0.65	0.64	0.64	247

Lampiran 12. Pengujian Menggunakan Data Baru



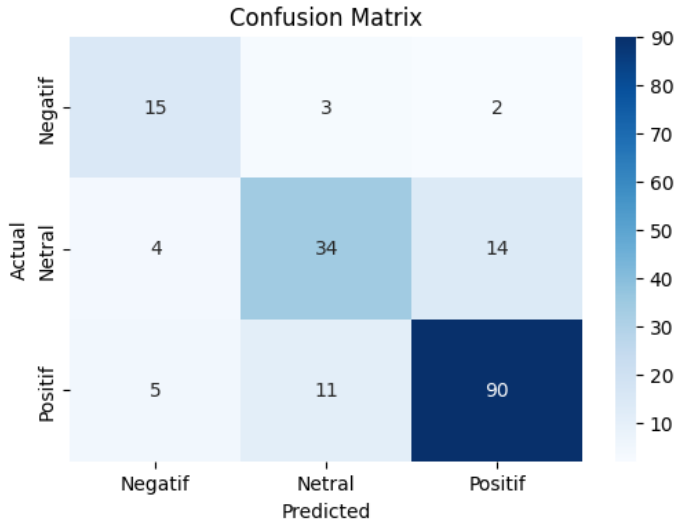
```
[1 1 2 2 0 1 0 0 0 2 1 2 1 1 2 1 1 2 2 1 1 1 1 0 2 1 1 1 1 1 1 2 1 1 1 2 0
 1 2 1 1 0 2 1 2 0 1 1 0 1 1 0 1 2 1 1 1 1 1 0 1 1 1 1 1 2 2 2 2 2 0 0 1 1 1
 1 0 1 0 2 1 1 1 0 2 2 2 2 1 2 2 2 2 2 0 0 2 2 1 1 1 2 2 1 1 1 2 0 2 2 0 1
 2 2 2 1 0 2 2 1 2 2 0 2 1 1 1 2 2 2 0 1 2 1 1 2 0 0 0 1 1 1 2 1 1 2 2 1 2
 2 0 1 1 2 1 1 1 2 2 2 1 0 2 2 2 2 2 2 2 0 1 2 0 1 2 2 1 1 1 2]
Akurasi: 65.17%
Confusion Matrix:
[[13  6  1]
 [ 6 41  5]
 [11 33 62]]
Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.65      0.52        20
           1       0.51      0.79      0.62        52
           2       0.91      0.58      0.71       106

    accuracy                           0.65       178
   macro avg       0.62      0.67      0.62       178
weighted avg       0.74      0.65      0.66       178
```

Lampiran 13. Hasil Evaluasi Model Dengan Max Feature 5000




```
Classification Report:
              precision    recall  f1-score   support

     Negatif       0.61      0.64      0.62        55
      Netral       0.43      0.35      0.39        66
     Positif       0.73      0.79      0.76       126

    accuracy                           0.64       247
   macro avg       0.59      0.59      0.59       247
weighted avg       0.63      0.64      0.63       247
```

Lampiran 14. v Hasil Evaluasi Model Dengan Max Feature 5000




```
Akurasi: 78.09%
Confusion Matrix:
[[15  3  2]
 [ 4 34 14]
 [ 5 11 90]]
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.75      0.68        20
           1       0.71      0.65      0.68        52
           2       0.85      0.85      0.85       106

    accuracy                           0.78       178
   macro avg       0.73      0.75      0.74       178
weighted avg       0.78      0.78      0.78       178
```

RIWAYAT HIDUP

Ni Luh Risma Pradnyadari lahir di Singaraja pada tanggal 14 Juli 2001. Penulis lahir dari pasangan suami istri Bapak I Ketut Mudarka dan Ibu Ni Luh Martini. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini penulis beralamat di Desa Suwug, Kec. Sawan, Kab. Buleleng, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD Negeri 1 & 2 Paket Agung dan lulus pada tahun 2013. Kemudian penulis melanjutkan di SMP Negeri 6 Singaraja dan lulus pada tahun 2016. Pada tahun 2019 penulis lulus dari SMK Negeri 3 Singaraja jurusan Teknik Komputer dan Jaringan. Selanjutnya, mulai tahun 2019 sampai dengan penulisan skripsi ini, penulis masih terdaftar sebagai mahasiswa Program S1 Ilmu Komputer di Universitas Pendidikan Ganesha.