# Technical Test Documentation

The general steps of this article classification problem solutions can be described as follows :

1. Preprocessing text data or article content.
2. Compare Tokenizer / Vectorizer (TFIDF and Doc2Vec) and Classifier ( MultinomialNB ,LogisticRegression, and Linear SVC). In order to validate The performance of each model, will use 5 fold cross-validation to check general performance of each model.
3. build the classifier with the best result tokenizer / vectorizer and classifier

## Preprocessing

The general idea of the preprocessing are described as follows:

1. Change all text into lower case. since Cat and cat is practically the same, we do not want Cat and cat become 2 different tokens when we do tokenize or vectorize.
2. Remove some trailing “\n” since there is some occurrences in the dataset
3. Filter out the word which only consisted alphabet and change it 0. The reasoning was because the numbers like year 1990 and 2001 will become 2 different token or vector although they means about the same. Furthermore we could not remove them altogether since number might be significant in some topics like economics but not in Lifestyle.
4. (Optional) Stem the word to basic its basic form (with the help of sastrawi libraries). However this step would need a long time to stem all the article content in the dataset. In the experiment, even with using multi processing libraries and **5 simultaneous process,the sastrawi stemmer would need around 45 minutes** to stem all the article content. In the next step we will compare the result with and without using sastrawi stemmer whether is it worth the time and resources used.

## Algorithm Selection

The Algorithm and Tokenize / Vectorizer that tried in this part are :

1. TFIDF vectorizer and Logistic Regression Classifier
2. TFIDF vectorizer and Multinomial Naive Bayes Classifier
3. TFIDF vectorizer and Linear SVC Classifier
4. Doc2Vec vectorizer and Multinomial Naive Bayes Classifier
5. Doc2Vec vectorizer and Linear SVC Classifier

Further the for each combination above we will compared the result using sastrawi stemmer and without it. Currently we would use default configurations for the algorithm and tokenizer / vectorizer except the following:

|  |  |  |
| --- | --- | --- |
| Type | Parameter | Value |
| SVM SVC | Class Weight | Balanced |
| Gensim Doc2Vec | Number Of Vector | 100 |
|  | Number Of Epoch | 30 |

In order to measure the performance of each combination, will use 5 fold cross-validation to check general performance of each combination and using 2 metric to measure which accuracy and balanced accuracy. Accuracy metric will measure general performance of each combinations and balanced accuracy will measure performance combinations dealing with imbalanced dataset since this article only has few data on some topic like shown below :

|  |  |
| --- | --- |
| Ekonomi | 1762 |
| Haji | 1497 |
| Hiburan | 1466 |
| Sepak Bola | 1184 |
| Internasional | 741 |
| Lifestyle | 572 |
| Teknologi | 571 |
| Sports | 435 |
| Bojonegoro | 260 |
| Kesehatan | 195 |
| Sains | 174 |
| Otomotif | 174 |
| Health | 131 |
| Politik | 104 |
| Hukum | 85 |
| Personal | 81 |
| Travel | 76 |
| Pendidikan | 70 |
| Sejarah | 70 |
| K-Pop | 61 |
| Obat-obatan | 58 |
| Horor | 50 |
| KPK | 37 |
| Regional | 35 |
| MotoGP | 35 |
| Bisnis | 25 |
| Pilgub Jatim | 25 |
| Keuangan | 14 |
| Jakarta | 12 |

The Result for each combination can be seen as below :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Vectorizer | Sastrawi Stemmer | Test Set  Average Accuracy | Test Set  Average Balanced Accuracy |
| Logistic Regression | TFIDF | No | 0.809829 | 0.390647 |
| Multinomial Naive Bayes | TFIDF | No | 0.67755 | 0.4160872 |
| SVM SVC | TFIDF | No | 0.87093 | 0.6650782 |
| Logistic Regression | TFIDF | Yes | 0.812638 | 0.399831 |
| Multinomial Naive Bayes | TFIDF | Yes | 0.668719 | 0.194731 |
| SVM SVC | TFIDF | Yes | 0.866612 | 0.673891 |
| Logistic Regression | Gensim Doc2Vec | No | 0.7825306 | 0.4242372 |
| SVM SVC | Gensim Doc2Vec | No | 0.7736872 | 0.5896652 |
| Logistic Regression | Gensim Doc2Vec | Yes | 0.787238 | 0.4242372 |
| SVM SVC | Gensim Doc2Vec | Yes | 0.769887 | 0.57321 |

From the results above, the best result would be using TFIDF, using sastrawi stemmer and Linear SVC Classifier. However, the gap with the second best result (TFIDF, no sastrawi stemmer and linear SVC) is quite insignificant but time and resources used it much more higher than second best result. Thus we will use the second best result (TFIDF, no sastrawi stemmer and linear SVC) as our solution for this technical test.

## Build Classifier and Solution

From the result of previous section we will use TFIDF without using sastrawi stemmer and Linear SVC classifier as our article classification solution. Below is the implementation corresponding with specified requirement by kumparan. There is comment or explanation for each of code in code snipplet of model.py below :

# insert necessary libraries needed for our models

from kumparanian import ds

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn import svm

from joblib import dump, load

class Model:

def \_\_init\_\_(self):

"""

You can add more parameter here to initialize your model

"""

# these block of codes below would load our stopword from 'id.stopwords.02.01.2016.txt' files and remove trailing '\n' character

with open('id.stopwords.02.01.2016.txt') as f:

self.stopword = f.readlines()

self.stopword = [line.rstrip('\n') for line in self.stopword]

# these block of codes below will initialize TFIDF and Linear SVC Classifier

self.tfidf\_vect = TfidfVectorizer()

self.clf = svm.LinearSVC(class\_weight='balanced')

def preprocessing(self,content):

"""

This method here is used to do preprocessing on article

"""

# Lines of code below will split the article into list of words

word\_split = content.split(' ')

# Lines of code below will remove trailing '\n' in the list

result = ['' for word in word\_split if '\n' in word]

# Lines of code below will change any word that non alphabet into 0 in the list

result = [word if word.isalpha() else 0 for word in word\_split]

# Lines of code below will remove any word in the list that found in stopword list

result = [word for word in result if word not in self.stopword]

# Lines of code below will join the list into single string

result = ' '.join(str(e) for e in result)

return result

def train(self):

"""

NOTE: Implement your training procedure in this method.

"""

# Lines of code below will read dataset into pandas dataframe

df = pd.read\_csv('data.csv')

# Lines of code below will remove any row with NA or empty cell in dataframe

df.dropna(inplace=True)

# Lines of code below will preprocessing article content into lower case (preprocessing step 1)

df.article\_content = df.article\_content.str.lower()

# Lines of code below will preprocessing artile content with the rest of preproessing step found in preprocessing method

df.article\_content = df.article\_content.apply(self.preprocessing)

# Lines of code below will build vector from dataset using TFIDF

self.tfidf\_vect.fit(df.article\_content)

# Lines of code below will transform article content into vector

train\_x\_tfidf = self.tfidf\_vect.transform(df.article\_content)

# Lines of code below will build linear SVC models based on vectorized article content

self.clf.fit(train\_x\_tfidf,df.article\_topic)

def predict(self, input):

"""

NOTE: Implement your predict procedure in this method.

"""

# Lines of code below will transform string input into vector. Note that since it expected list or array as input

# thus we gonna convert the input into [input] or list with single element of our input

input\_vect = self.tfidf\_vect.transform([input])

# Lines of code below will predict the topic based from vectorized input

result = self.clf.predict(input\_vect)

# since we inputted a list of input then the output result will be list of topic,

# hus we only need return first element of result since we only input single element of input

return result[0]

def save(self):

"""

Save trained model to model.pickle file.

"""

ds.model.save(self, "model.pickle")

if \_\_name\_\_ == '\_\_main\_\_':

# NOTE: Edit this if you add more initialization parameter

model = Model()

# Train your model

model.train()

# Save your trained model to model.pickle

model.save()

The 'id.stopwords.02.01.2016.txt' files already attached inside the zip files. In order to run the classifier models, just run “python model.py” (please make sure you already have libraries needed as shown in import)

## Future Improvement

Few improvements that we could do to further improve our classifier performance are :

1. Improving stopwords dictionary especially stopwords for specific articles.
2. Build our own stemmer or lemmatization which either make from scratch or optimizing existing libraries like sastrawi stemmer.
3. Increasing dataset especially for topic that has low count article sample. I believe this will significantly improve our models classifier accuracy. For example, below is the single classification result of training with TFIDF, without sastrawi and Linear SVC. the row with yellow and red colors are topic with bad result. Note that almost all of them are topic with low count of article sample. Thus with this in mind, i strongly believe that we could increase our model classifier result significantly with increasing their sample article

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Bisnis | 1 | 0.33 | 0.5 | 6 |
| Bojonegoro | 0.83 | 0.91 | 0.87 | 43 |
| Ekonomi | 0.91 | 0.96 | 0.94 | 357 |
| Haji | 0.98 | 0.98 | 0.98 | 306 |
| Health | 0.39 | 0.36 | 0.37 | 25 |
| Hiburan | 0.9 | 0.96 | 0.93 | 284 |
| Horor | 0.86 | 0.67 | 0.75 | 9 |
| Hukum | 0.72 | 0.68 | 0.7 | 19 |
| Internasional | 0.86 | 0.88 | 0.87 | 147 |
| Jakarta | 1 | 0.5 | 0.67 | 2 |
| K-Pop | 1 | 0.67 | 0.8 | 6 |
| KPK | 1 | 0.4 | 0.57 | 5 |
| Kesehatan | 0.63 | 0.57 | 0.6 | 46 |
| Keuangan | 0 | 0 | 0 | 3 |
| Lifestyle | 0.8 | 0.88 | 0.84 | 119 |
| MotoGP | 0.6 | 0.5 | 0.55 | 6 |
| Obat-obatan | 0.6 | 0.43 | 0.5 | 7 |
| Otomotif | 0.94 | 0.89 | 0.92 | 37 |
| Pendidikan | 1 | 0.77 | 0.87 | 13 |
| Personal | 0.57 | 0.21 | 0.31 | 19 |
| Pilgub Jatim | 0.43 | 0.6 | 0.5 | 5 |
| Politik | 0.59 | 0.53 | 0.56 | 19 |
| Regional | 0.67 | 0.29 | 0.4 | 7 |
| Sains | 1 | 0.81 | 0.9 | 27 |
| Sejarah | 0.62 | 0.36 | 0.45 | 14 |
| Sepak Bola | 0.83 | 0.95 | 0.88 | 241 |
| Sports | 0.75 | 0.52 | 0.61 | 104 |
| Teknologi | 0.88 | 0.89 | 0.89 | 102 |
| Travel | 0.73 | 0.53 | 0.62 | 15 |
|  |  |  |  |  |
| accuracy | 0.871993 | | | |
| macro avg | 0.76 | 0.62 | 0.67 | 1993 |
| weighted avg | 0.86 | 0.87 | 0.86 | 1993 |