Hands On Classification & Clustering

Hadaiq Rolis Sanabila

hadaiq@cs.ui.ac.id

Natural Language Processing and Text Mining

Pusilkom UI
22 – 26 Maret 2016

pusilkom ui
quality results

CLASSIFICATION USING NLTK

Preparation

- Donwload nltk.corpus -> names
  - Go to python interpreter on CMD
    - import nltk
    - nltk.download()
    - choose names corpora
    - find: C:\Users\MIC-UI\AppData\Roaming\nltk_data\corpora

Classification

- Prepare the feature function:

```
def gender_features(word):
        features = {};
        features['last_letter'] =word[-1];
        return features;
```

Classification (cont.)

- Prepare the corpus:

```
from nltk.corpus import names;

labeled_names = ([(name, 'male') for name in
names.words('male.txt')] + [(name, 'female') for name in
names.words('female.txt')]);

#print labeled_names;
```

Classification (cont.)

- Shuffle the data (avoid bias):

```
import random;
random.shuffle(labeled_names);
```

Classification (cont.)

- Divide the training set and test set:

```
from nltk.classify import apply_features

train_set = apply_features(gender_features,
labeled_names[500:]);

test_set = apply_features(gender_features, labeled_names[:500]);

#print test_set;
```

Classification (cont.)

- Train the classifier

```
import nltk;
classifier = nltk.NaiveBayesClassifier.train(train_set);
```

Classification (cont.)

- Test the classifier

```
print classifier.classify(gender_features('Neo'));
print classifier.classify(gender_features('Trinity'));
print(nltk.classify.accuracy(classifier, test_set));
classifier.show_most_informative_features(5)
```

Classification Using Own Dataset

- Copy dataset (folder “kompas”) to
  - C:\Users\MIC-UI\AppData\Roaming\nltk_data\corpora
- Check the dataset:
  - nltk.data.find('corpora/kompas')
- Donwload nltk.corpus -> stopwords
  - copy file: stop_indo to:
    - C:\Users\MIC-UI\AppData\Roaming\nltk_data\corpora\stopwords

pusilkom ui
quality results

Classification Using Own Dataset (Cont.)

- Import the required library

```
import string
from itertools import chain
from nltk.corpus import stopwords
from nltk.probability import FreqDist
from nltk.classify import NaiveBayesClassifier as nbc
from nltk.corpus import CategorizedPlaintextCorpusReader
import nltk
```

Classification Using Own Dataset (Cont.)

- Prepare the corpus:

```
mydir = 'C:/Users/MIC-UI/AppData/Roaming/nltk_data/corpora/kompas'

mr = CategorizedPlaintextCorpusReader(mydir, r'(?!\.).*\.txt',
cat_pattern=r'(ekonomi|kesehatan|olahraga|properti|travel)/.*',
encoding='ascii')

stop = stopwords.words('stop_indo')

documents = [([w for w in mr.words(i) if w.lower() not in stop and
w.lower() not in string.punctuation], i.split('/')[0]) for i in
mr.fileids()]
```

Classification Using Own Dataset (Cont.)

- Select the features

```
word_features = FreqDist(chain(*[i for i,j in documents]))

word_features = word_features.keys()[:100]
```

Classification Using Own Dataset (Cont.)

- Divide the training set and test set:

```
numtrain = int(len(documents) * 90 / 100)

train_set = [({i:(i in tokens) for i in word_features}, tag) for
tokens,tag in documents[:numtrain]]

test_set = [({i:(i in tokens) for i in word_features}, tag) for
tokens,tag  in documents[numtrain:]]
```

Classification Using Own Dataset (Cont.)

- Train and test the classifier:

```
classifier = nbc.train(train_set)

print nltk.classify.accuracy(classifier, test_set)

classifier.show_most_informative_features(5)
```

Task

Update your model (especially on the feature selection process to get the higher precission & recall)

Hints:

1. Lower-casing: The entire email is converted into lower case, so that captialization is ignored (e.g., IndIcaTE is treated the same as Indicate).
2. Stripping HTML: All HTML tags are removed from the emails. Many emails often come with HTML formatting; we remove all the HTML tags, so that only the content remains.
3. Normalizing URLs: All URLs are replaced with the text “httpaddr”.
4. Normalizing Email Addresses: All email addresses are replaced with the text “emailaddr”.
5. Normalizing Numbers: All numbers are replaced with the text “number”.
6. Word Stemming: Words are reduced to their stemmed form. For example, “discount”, “discounts”, “discounted” and “discounting” are all replaced with “discount”.
7. Removal of non-words: Non-words and punctuation have been removed. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character.
8. Try to get representative words from each class as features

pusilkom ui
quality results

CLUSTERING USING NLTK

Preparation

- Copy dataset (folder “kompas_clustering”) to
  - C:\Users\MIC-UI\AppData\Roaming\nltk_data\corpora
- Check the dataset:
  - nltk.data.find('corpora/kompas_clustering')

Clustering

- Prepare the function

```
def process_text(text, stem=False):
    tokens = word_tokenize(text)
    return tokens
```

Clustering (cont.)

- Prepare the required library

```
import string
import collections

from nltk import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from pprint import pprint
```

Clustering (cont.)

- Prepare the clustering function

```
def cluster_texts(texts, clusters=3):
    vectorizer = HashingVectorizer(tokenizer=process_text,
                                   stop_words=stopwords.words('stop_indo'),
                                   lowercase=True)
    tfidf_model = vectorizer.fit_transform(texts)
    km_model = KMeans(n_clusters=clusters, n_init = 100)
    km_model.fit(tfidf_model)

    ...
```

Clustering (cont.)

- Prepare the clustering function

```
def cluster_texts(texts, clusters=3):
    ...
    clustering = collections.defaultdict(list)

    for idx, label in enumerate(km_model.labels_):
        clustering[label].append(idx)

    return clustering
```

Clustering (cont.)

- Prepare the Main function

```
if __name__ == "__main__":
        contents = list();
        import os
        rootdir = 'C:/Users/…/nltk_data/corpora/kompas_clustering/'
        for subdir, dirs, files in os.walk(rootdir):
                for file in files:
                        f=open(rootdir+file,'r')
                        lines = f.readlines()
                        f.close()
                        tmp = str();
                        for line in lines:
                                tmp += line.strip() + " ";
                        contents.append(tmp)
        articles = contents
        clusters = cluster_texts(articles, 5)
        pprint(dict(clusters))
```

Task

Update your model to get better cluster

Hints:

1. Lower-casing: The entire email is converted into lower case, so that captialization is ignored (e.g., IndIcaTE is treated the same as Indicate).
2. Stripping HTML: All HTML tags are removed from the emails. Many emails often come with HTML formatting; we remove all the HTML tags, so that only the content remains.
3. Normalizing URLs: All URLs are replaced with the text “httpaddr”.
4. Normalizing Email Addresses: All email addresses are replaced with the text “emailaddr”.
5. Normalizing Numbers: All numbers are replaced with the text “number”.
6. Word Stemming: Words are reduced to their stemmed form. For example, “discount”, “discounts”, “discounted” and “discounting” are all replaced with “discount”.
7. Removal of non-words: Non-words and punctuation have been removed. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character.

Thank you