

# Spam mail classification using Natural Language Processing

Hao Zhou(001832050) Huahui Chao(001817189)

**Abstract:** This project report illustrates how the technology in Natural Language Processing work in a real-world problem: spam mail classification[1]. Using some important ways of the text preprocessing technology like Noise removal, Tokenization, Normalization all make the text we want more related to the mail theme as to classify its attribute. We also use different machine learning algorithm to compare the classification report and then get all related score to test the fitness to the models. In this project we pay more attention to the preprocessing of the text inside the mail and the choice of machine learning algorithm in order to get most related words as to fit the model without too many noises.

## 1.Introduction

With the use of computer and the internet become essential to the world, the machine can also start to understand Human language, and when Artificial Intelligence and machine learning comes in our life, we start to think about the detection of different language of humans, that's where Natural Language Processing comes in. The history of NLP dates back to the seventeenth century, when philosophers such as Leibniz and Descartes put forward proposals for codes which would relate words between languages. But all those ones remained theoretical [2]. And now the NLP becomes a power area can help people in many ways like spellcheck and autocorrect, auto-generated video captions, virtual assistants like Amazon's Alexa, email's suggested replies and your phone's text prediction. This project report illustrates how the technology in Natural Language Processing work in a real-world problem: spam mail classification. Using some important ways of the text preprocessing technology like Noise removal, Tokenization, Normalization all make the text we want more related to the mail theme as to classify its attribute. We want to use different machine learning algorithm and different preprocessing technology to research how the different choice of these issues effect the robust of the model.

## 2.Research related to NLP

### Text preprocessing

1.Noise removal --- stripping any text formatting (eg. HTML tags)

2. Tokenization --- breaking text into individual words

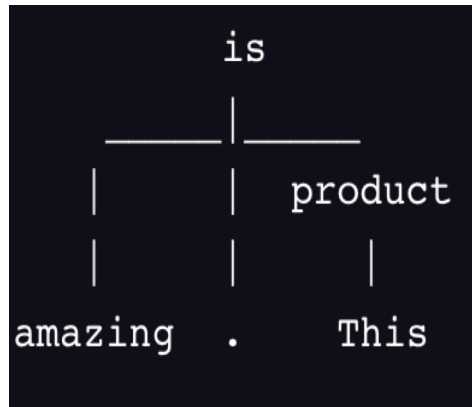
2.1 Parsing text : concern with segmenting text based on syntax

NLTK:

1). Part-of-speech tagging (POS tagging): NLTK can identify part of text in the tag of verbs, nouns, adjectives fast and accurately.

2). Named entity recognition (NER): capture the topic of text, help to identify the proper nouns

3). Dependency grammar trees: help people understand the relationship of the words in sentence.  
eg. "This product is amazing.(fig1. Dependency grammar trees)



1. Dependency grammar trees

4). Regex parsing: using Python's re library, when coupled with POS tagging, you can identify specific phrase chunks.

3. Normalization --- almost anything else you'd do to clean up your text data:

3.1 Stemming: cut word prefixes and suffixes. "starting" and "started" change to "start", but "sing" may become "s" and "sung" would remain "sung."

3.2 Lemmatization: bring words down to their root form. Eg. "am" and "are" are related to "be." eg.

playing, plays, played → Play

am, is, are → be

car, cars, car's, cars' → car

Using above mapping a sentence can be normalized as below:

the product's material is better than others. → the product material be good than other.

### Compare between stemming and lemmatization

text	Stemmed text	Lemmatized text
so	so	so
many	mani	many
are	are	be
jumping	jump	jump
suitcases	suitcas	suitcase
tightly	tightli	tightly
went	went	go
arriving	arriv	arrive
noticed	notic	notice
barely	bare	barely
the	the	the
anywhere	anywher	anywhere
hardly	hardli	hardly
dentist's	dentist, s	dentist, s

Lemmatization is different from Stemming, it changes the words properly in the form of their root word belongs to the language. Stem might not be an actual word, but lemmatization is an actual word. Stemming follows an algorithm to perform words faster. However, lemmatization produce a word slower than stemming, it need to define a set of words in correct form.

Code:

## 1. Stemming

```
def text_process_stem(text):
    """
    1.tokenize
    2.stemmenize
    3.remove stopwords
    4.return cleaned text words
    """
    nonpunc = [char for char in text if char not in string.punctuation]
    nonpunc = ''.join(nonpunc)
    stemmer = PorterStemmer()
    stemmed = [stemmer.stem(token) for token in nonpunc.split()]
    stopword_remove = [word for word in stemmed if word.lower() not in stopwords.words('english')]
    return stopword_remove
```

## 2. lemmatization

This is part-of-speech(POS) method must be defined when using lemmatization

```
from nltk.corpus import wordnet
from collections import Counter
def get_part_of_speech(word):
    probable_part_of_speech = wordnet.synsets(word)
    pos_counts = Counter()
    pos_counts["n"] = len([ item for item in probable_part_of_speech if item.pos()=="n" ])
    pos_counts["v"] = len([ item for item in probable_part_of_speech if item.pos()=="v" ])
    pos_counts["a"] = len([ item for item in probable_part_of_speech if item.pos()=="a" ])
    pos_counts["r"] = len([ item for item in probable_part_of_speech if item.pos()=="r" ])

    most_likely_part_of_speech = pos_counts.most_common(1)[0][0]
    return most_likely_part_of_speech
```

```
def text_process_lemma(text):
    """
    1.tokenize
    2.lemmatizer
    3.remove stopwords
    4.return cleaned text words
    """
    nonpunc = [char for char in text if char not in string.punctuation]
    nonpunc = ''.join(nonpunc)
    lemmatizer = WordNetLemmatizer()
    lemmatized = [lemmatizer.lemmatize(token, get_part_of_speech(token)) for token in nonpunc.split()]
    stopword_remove = [word for word in lemmatized if word.lower() not in stopwords.words('english')]
    return stopword_remove
```

### 3.3 lowercasing, punctuation removal, stopwords removal, spelling correction, etc.

Stopwords: These words do not contain important significant meaning in search queries. We need to filter these words from search queries, because they will return a large amount of unnecessary information. In English words such as ‘the, be, are, as’ etc.

## Language Model

### 1. Language prediction: Language Model – Bag of Words Approach

1.1 We can help computers make predictions about language by training a language model on a corpus. We build and use language models (probabilistic computer models of language) to figure out the possibility of letter, word, or phrase will be used. One of the most common language models is named bag of words. This model count of each word in this sentence.

Eg. “The material of the product is amazing.”

After some preprocessing, bag of words would result in a mapping as below:

{“the”: 2, “material”: 1, “of”: 1, “product”: 1, “is”: 1, “amazing”: 1}

Despite of the different word order and sentence structure, Bag of words can be a great way for us to make prediction about text theme or sentiment.

1.2 In the project, we will convert each message into a vector the Scikit Learn’s algorithm models can work with.

Bag of words model:

1).Count how many times does a word occur in each message.

2).Weight the counts, so that frequent tokens get lower weight.

3).Normalize the vectors to unit length, to abstract from original text length.

This model will convert a collection of text documents to a matrix of token counts. It will be surely for the scikit learn to output a sparse matrix.

	Message 1	Message 2	...	Message N
Word 1 Count	0	1	...	0
Word 2 Count	0	0	...	0
...	1	2	...	0
Word N Count	0	1	...	1

2.Vectors in Bag of words

### 1.3 Addressing text similarity & spelling correction

To solving the problem of text similarity and spelling correction, we will consider about the Levenshtein distance or minimal edit distance between two words. This method is calculation the number of insertions, deletions, and substitutions between two words. Spelling correction need consider key distance on keyboard and technology of phonetic similarity.

eg. “amusing” & “amazing”, when “amusing” turn to “amazing”, it need “u → a, s → z”, two times substitution, so the Levenshtein distance would be 2.

### 2. Attention to each word’s neighbors: Language Model – N- Gram and NLM

N-gram model considers a sequence of some number (n) units and calculates the probability of each unit in text. The larger n values can be impressive at language prediction.

eg. “This product is amazing. This product has good quality.”

The neighbors in these sentences count probability:

{(‘,’,’this’) : 2, (‘this’,’product’) : 2, (‘product’,’is’) : 1, (‘is’,’amazing’) :1, (‘amazing’, ‘’) : 1, (‘product’,’has’) : 1, (‘has’,’good’) : 1, (‘good’,’quality’) : 1, (‘quality’,’’) : 1}

### Topic Model

#### 1.Term frequency-inverse document frequency (tf-idf)

1.1 Term Frequency: importance of the term in that document ( $tf_{x,y}$ )

1.2 Inverse Document Frequency: important of the term in corpus ( $\log(\frac{N}{df_x})$ )

1.3 In mathematical way, tf-idf is express as below:

$$W_{x,y} = tf_{x,y} * \log(\frac{N}{df_x})$$

Term x within document y,  $tf_{x,y}$  = frequency of x in y,  $df_x$  = number of documents containing x, N = total number of documents

In the project, we found the answer of the question of ‘Why would you want to give more priority to less-used words?’, the reason is when we figure out a large amount of text, we found high probability in text are the words like “the”, “is”, “a”. If you want to determine your topic obviously, you need to use Python libraries genism and sklearn have modules to handle tf-idf.

## 2. Word2vec

Word2vec is technique can map the topic like vectors that similar words in text will bond together. Word2vec is a two-layer neural net that processes text. Its input is text corpus and output is vectors: word-to-vector feature. While Word2vec is not a deep neural network, It turns text into a numerical form that deep nets can understand.[3]

## 3. Machine learning algorithm in the project

In this project, we want to compare the difference between stem method and lemmatization method to process the data. And this is considered to be a key step to decide whether the process of the data is appropriate.

First thing is to create a base model, which means we directly tokenize the word and remove the punctuations in all of the E-mails, and we use machine learning algorithm to train and test the model, and then we will add stem and lemmatization method to get different outcome.

Naïve Bayes Algorithm:

Base model:

```

from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=base_process)),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

from sklearn.metrics import classification_report
print(classification_report(label_test, pred))

```

	precision	recall	f1-score	support
ham	0.93	1.00	0.97	2412
spam	1.00	0.55	0.71	374
avg / total	0.94	0.94	0.93	2786

Lemmatization method:

## Naive Bayes Algorithm

```

from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_lemma)),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

from sklearn.metrics import classification_report
print(classification_report(label_test, pred))

```

	precision	recall	f1-score	support
ham	0.95	1.00	0.98	1439
spam	1.00	0.69	0.82	233
avg / total	0.96	0.96	0.95	1672

Stem method:

```

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_stem)),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))

```

	precision	recall	f1-score	support
ham	0.95	1.00	0.98	1439
spam	1.00	0.70	0.82	233
avg / total	0.96	0.96	0.95	1672

## 1. GBM Algorithm:

Base model:

```

from sklearn.ensemble import GradientBoostingClassifier

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=base_process)),
    ('tfidf', TfidfTransformer()),
    ('classifier', GradientBoostingClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))

```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	1448
spam	1.00	0.72	0.84	224
avg / total	0.96	0.96	0.96	1672

Lemmatization method:

## GBM Algorithm

```
from sklearn.ensemble import GradientBoostingClassifier

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_lemma)),
    ('tfidf', TfidfTransformer()),
    ('classifier', GradientBoostingClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.95	1.00	0.98	1439
spam	0.98	0.70	0.82	233
avg / total	0.96	0.96	0.95	1672

Stem method:

```
pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_stem)),
    ('tfidf', TfidfTransformer()),
    ('classifier', GradientBoostingClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	1439
spam	0.97	0.75	0.84	233
avg / total	0.96	0.96	0.96	1672



## 2. Random Forest Algorithm:

Base model:

```
from sklearn.ensemble import RandomForestClassifier

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=base_process)),
    ('tfidf', TfidfTransformer()),
    ('classifier', RandomForestClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	1448
spam	1.00	0.74	0.85	224
avg / total	0.97	0.97	0.96	1672

Lemmatization method:

## Randomforest Algorithm

```
from sklearn.ensemble import RandomForestClassifier

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_lemma)),
    ('tfidf', TfidfTransformer()),
    ('classifier', RandomForestClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	1439
spam	1.00	0.75	0.86	233
avg / total	0.97	0.96	0.96	1672

Stem method:

```
pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_stem)),
    ('tfidf', TfidfTransformer()),
    ('classifier', RandomForestClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	1439
spam	0.99	0.72	0.83	233
avg / total	0.96	0.96	0.96	1672

3. Logistic Regression:

Base model:

```
from sklearn.linear_model import LogisticRegression
pipeline = Pipeline([
    ('baw', CountVectorizer(analyzer=base_process)),
    ('tfidf', TfidfTransformer()),
    ('model', LogisticRegression()),
])
pipeline.fit(mil_train, label_train)
pred = pipeline.predict(mil_test)
#Evaluation
print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	1448
spam	1.00	0.70	0.82	224
avg / total	0.96	0.96	0.96	1672

Lemmatization method:

## Logistic Regression

```
from sklearn.linear_model import LogisticRegression
pipeline = Pipeline([
    ('baw', CountVectorizer(analyzer=text_process_lemma)),
    ('tfidf', TfidfTransformer()),
    ('model', LogisticRegression()),
])
pipeline.fit(mil_train, label_train)
pred = pipeline.predict(mil_test)
#Evaluation
print(classification_report(label_test, pred))
```

	precision	recall	f1-score	support
ham	0.95	1.00	0.97	1439
spam	0.99	0.67	0.80	233
avg / total	0.95	0.95	0.95	1672

Stem method:

```

: pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process_stem)),
    ('tfidf', TfidfTransformer()),
    ('classifier', RandomForestClassifier())
])

pipeline.fit(mil_train, label_train)

pred = pipeline.predict(mil_test)

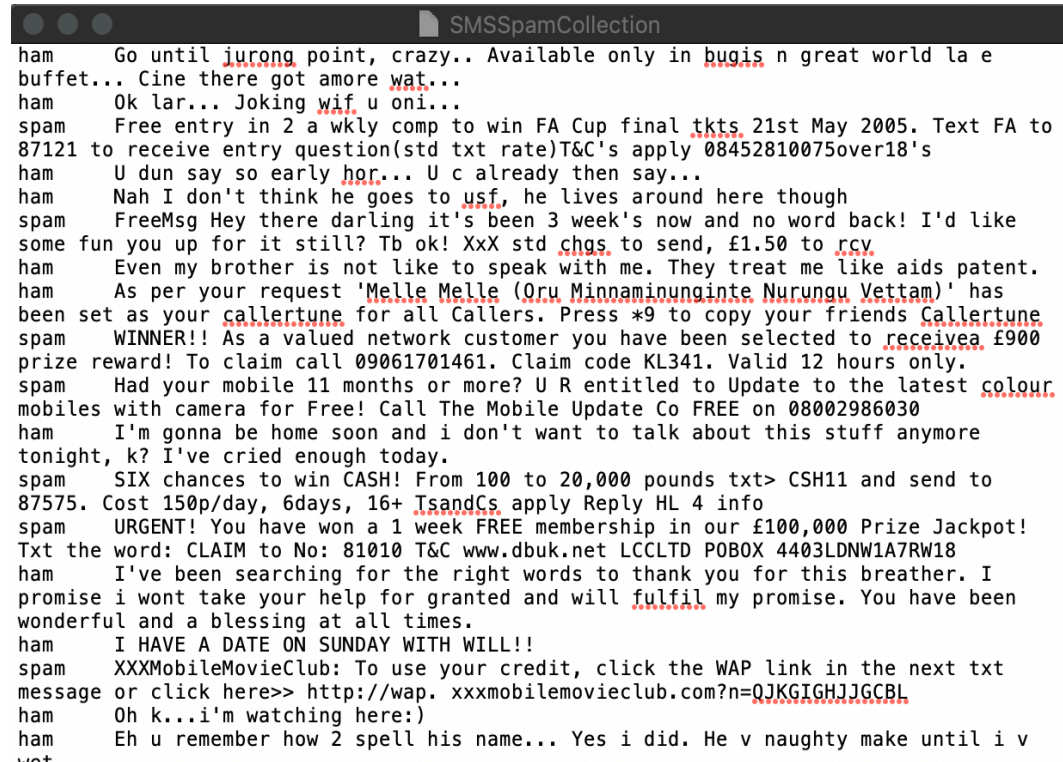
print(classification_report(label_test, pred))

```

	precision	recall	f1-score	support
ham	0.95	1.00	0.98	1439
spam	0.99	0.70	0.82	233
avg / total	0.96	0.96	0.96	1672

## 4. Data source

Our data comes from UCI Machine Learning Repository, it is called SMS Spam Collection Data Set.



```

SMSSpamCollection
ham    Go until jurong point, crazy.. Available only in bugis n great world la e
buffet... Cine there got amore wat...
ham    Ok lar... Joking wif u oni...
spam   Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to
87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham    U dun say so early hor... U c already then say...
ham    Nah I don't think he goes to usf, he lives around here though
spam   FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like
some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
ham    Even my brother is not like to speak with me. They treat me like aids patent.
ham    As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has
been set as your callertune for all Callers. Press *9 to copy your friends Callertune
spam   WINNER!! As a valued network customer you have been selected to receive a £900
prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
spam   Had your mobile 11 months or more? U R entitled to Update to the latest colour
mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
ham    I'm gonna be home soon and i don't want to talk about this stuff anymore
tonight, k? I've cried enough today.
spam   SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to
87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
spam   URGENT! You have won a 1 week FREE membership in our £100,000 Prize Jackpot!
Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
ham    I've been searching for the right words to thank you for this breather. I
promise i wont take your help for granted and will fulfil my promise. You have been
wonderful and a blessing at all times.
ham    I HAVE A DATE ON SUNDAY WITH WILL!!
spam   XXXMobileMovieClub: To use your credit, click the WAP link in the next txt
message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
ham    Oh k...i'm watching here:)
ham    Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v
wat

```

#### **4. Challenge and Considerations**

This model is only for English language, I think the difficult is that even if the English self has many new added words and maybe unrecognizable words with dialect and accent. And then the difficult of NLP may differs a lot.

And then in terms of different language, we don't have that much libraries to prepare for language I think that is also another challenge to complete NLP.

#### **Reference**

1. Rebecca J. Passonneau, Cynthia Rudin, Axinia Radeva, and Zhi An Liu: Reducing Noise in Labels and Features for a Real World Dataset: Application of NLP Corpus Annotation Methods. Columbia University, New York, NY 10027, USA
2. [https://en.wikipedia.org/wiki/History\\_of\\_natural\\_language\\_processing](https://en.wikipedia.org/wiki/History_of_natural_language_processing)
3. <https://skymind.ai/wiki/word2vec>