

Spam Message Filter

Fasttext와 CNN을 사용한 binary classification

기계학습(COSE 362-02)

컴퓨터학과

2015130741 원혜진

INDEX

1. 요약	3
2. 프로젝트 개요 및 primitives.....	3
3. 프로그램 구성 및 세부사항	3
4. Train 모델 비교	6
5. 실행 방법	7
6. 사용 library.....	8
7. 결론 및 제언.....	9

1. 요약

- 본 프로젝트는 이메일 내 불필요한 스팸메일을 구분하기 위해 진행되었다. 스팸 여부가 표시되어 있는 메일 데이터를 사용하였으며, fasttext와 CNN을 사용하여 binary classification을 진행하였다.

- 가장 성능이 좋은 CNN 모델은 2-gram~6-gram을 사용한 train 모델 (11050156_100_0.0512.hdf5)로 test accuracy(leader board data 사용)은 **94.979%**의 성능을 보였다.

test_size	val_size	test_size	epochs	batch_size	early stopping patience
0.8	0.2	0.1	100	128	30

[표1_11050156_100_0.0512.hdf5 모델의 hyperparameters]

2. 프로젝트 개요 및 primitives

- Spam filter classification은 binary classification에 해당하며, 스팸 메일이라면 1, 스팸 메일이 아니라면 0으로 분류한다.

- Spam Filter Classifier가 갖춰야 하는 기능은 아래와 같다.

1) 원본 데이터의 'message'를 word vector형태로 변환 : message를 CNN의 feature로 사용하기 위해서는 벡터로 표현되어야 하므로, 변환이 필요하다. 데이터 확인 시, 구어체의 단어가 많이 분포하고 있어, OOV(Out of Vocabulary)의 가능성이 높다. 대부분의 구어체는 핵심 형태소와 오타가 결합된 형태로, 구어체와 문어체의 일부분이 일치하는 특성이 있다.(e.g. 구어체 hellow와 문어체 hello) 따라서, 단어의 n-gram(n개의 character로 이루어진 group)을 기반으로 학습한 **fasttext**를 사용하였다.

2) 각 sample 별 binary classification : document의 경우, 단순한 단어의 분포로 분류하는 데에는 어려움이 있는데, 이는 같은 단어일지라도 다양한 class에서 사용될 수 있기 때문이다. 따라서, 문맥을 반영할 수 있는 모델이 적합하므로, [Convolutional Neural Networks for Sentence Classification](#) (EMNLP 2014) 논문을 참고하여 **CNN 모델을 구현**하였다.

3. 프로그램 구성 및 세부사항

- spamFilter.py는 1) main과 2) class SpamFilter()와 두가지 부분으로 이루어져 있다.

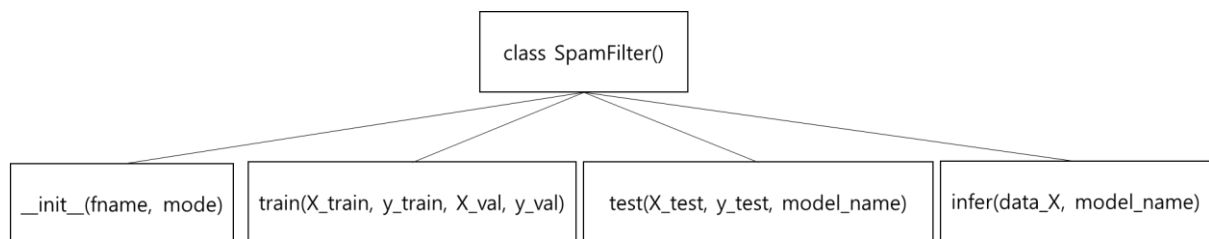
3-1. main

- 사용자가 입력한 모드에 따라, Train, Test, Infer 기능을 수행할 수 있다.

Mode Number	Mode	Description
0	Train	새로운 classification model을 학습함
1	Test	모델 훈련시 분리한 test data를 사용하여 accuracy 측정
2	Infer	label이 없는 데이터와 pre-trained model을 사용하여 class 예측

[표2_프로그램 실행 mode]

3-2. class SpamFilter()



* cleanTxt(string) 함수는 특수문자/한글 등의 처리를 위한 부차 함수이므로 생략

[그림1_SpamFilter 구조]

3-2-1. __init__(fname, mode)

- Parameters

- fname: sample data의 파일명
- mode: 0, 1, or 2

- Description : fname 내의 데이터를 전처리하는 함수.

- ① Text cleaning: message 내 영어 및 특수문자를 제한 문자를 삭제(e.g. 첼릿), 약어 및 특수문자를 하나의 단어로 취급하기 위해 앞에 띄어쓰기 삽입(e.g. '!' -> ' !')
- ② Tokenizing: token이란, 문장을 이루는 단위로, 띄어쓰기로 구분됨. 토큰 단위로 벡터화하기 위해, message를 띄어쓰기 단위로 split
- ③ Word vectorizing: 각 message를 fasttext 모델인 'wordvec_model/cc.en.300.bin'을 사용하여 word vector로 변환
- ④ Padding & Augmentation: message별 token의 수가 다르므로, 고정된 CNN Input 크기에 맞추어 zero padding을 추가해야 함. Padding이 추가될 위치를 앞/뒤 중에 정할 수 있다. zero padding + data와 data+zero padding은 다른 data

로, 1개의 sample data를 통해 2개의 sample data를 생성하였다.

```

45 if mode == 0 or mode == 1:
46     self.messages = pad_sequences(self.messages, dtype='float32', maxlen=191, padding='pre')
47     self.messages_aug = pad_sequences(self.messages, dtype='float32', maxlen=191, padding='post')
48     self.messages = np.concatenate((self.messages, self.messages_aug), axis=-1)
49 else:
50     self.messages = pad_sequences(self.messages, dtype='float32', maxlen=191, padding='pre')

```

[그림2_augmentation code]

1	['Hello',	→	[[[0.1, 0.8, 0.009....0.0001],
	'I',		[0.2, 0.7, 0.009....0.0001],
	'am']		[0.05, 0.008, 0.009....0.01]]
2	['this',	→	[[[0.1, 0.8, 0.009....0.0001],
	'is',		[0.2, 0.7, 0.009....0.0001],
	'second']]		[0.05, 0.008, 0.009....0.01]]]

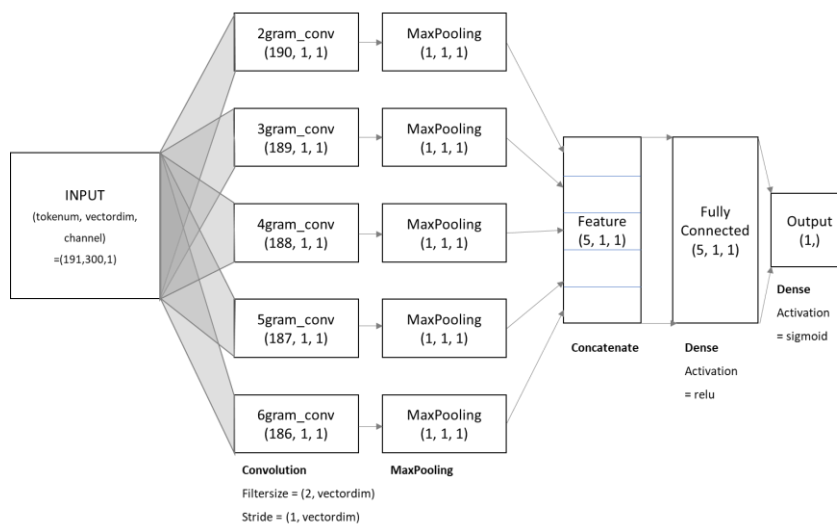
[표3_word vector 변환 예시]

3-2-2. train(X_train, y_train, X_val, y_val)

- Parameters

- X_train, y_train: train을 위한 data와 label. X_train.shape, y_train.shape= (?, 191, 300, 1), (1,)
- X_val, y_val: validation을 위한 data와 labe. X_val.shape, y_val.shape= (?, 191, 300, 1), (1,)

- Description



[그림2_Spam Filter CNN 모델 구조]

- Input layer
 - tokennum(191): Tokennum은 sample data 내 message별 token의 수를 비교하여 최대 token의 수로 설정되었다.
 - vectordim(300): fastext 모델 word vector의 차원수
- Convolution layer
- MaxPooling layer
- Feature map layer
- Fully Connected layer
 - 이전 layer인 Feature map layer는 max pooling의 결과로, 높은 값으로 이루어질 가능성이 높다. Activation 함수로 sigmoid를 사용할 경우, 역전파에 어려움이 있을 수 있으므로 ReLU를 사용하였다.
- Output
 - 1/0을 구분해야 하므로, activation 함수로 'sigmoid'를 사용하였다.

3-2-3. test(X_test, y_test, model_name)

- Parameters

- X_test, y_test: test를 위한 data와 label. X_train.shape, y_train.shape= (?, 191, 300, 1), (1,)
- model_name: test할 때 사용하고자 하는 모델(.hdf5)

3-2-4. infer(data_X, model_name)

- Parameters

- data_X : infer하고자 하는 데이터. data_X.shape = (?, 191, 300, 1)
- model_name: infer할 때 사용하고자 하는 모델(.hdf5)

4. Train 모델 비교

- n-gram의 범위는 2~6이 2~5보다 accuracy가 높다.
- augmentation을 진행할 경우, validation accuracy는 높아졌으나, leaderboard accuracy는 높아졌


다. 앞/뒤 zero padding 2개 버전을 모두 사용하여 augmentation을 진행했으나, 정의한 CNN 구조상, max pooling layer에서는 2개 버전이 유사하게 나오고, 이로 인해 overfitting이 발생한 것으로 추정된다.

Model name	epochs	min_n-gram	max_n-gram	early stopped	augmentation	min val loss	leaderboard accuracy
50-0.0823.hdf5	50	2	5	50	X	0.0823	0.86624
11050123_050_0.0665.hdf5	50	2	5	50	X	0.06	0.9000
11050151_058_0.0262.hdf5	100	2	6	58	X	0.0262	0.94168
11050156_100_0.0512.hdf5	100	2	6	100	X	0.0512	0.94797
11050212_053_0.0156.hdf5	100	2	6	53	O	0.0156	0.91463
11050219_037_0.0159.hdf5	100	2	6	37	O	0.0159	0.92215

5. 실행 방법

0. 준비 사항: 'wordvec_model' 디렉토리에 'cc.en.300.bin.gz'파일(fasttext word vector model) 다운로드 후 압축 해제

- <https://fasttext.cc/docs/en/crawl-vectors.html> 접속


[Docs](#)
[Resources](#)
[Blog](#)
[GitHub](#)

Models

The models can be downloaded from:

Afrikaans: bin, text	Albanian: bin, text	Alemannic: bin, text
Amharic: bin, text	Arabic: bin, text	Aragonese: bin, text
Armenian: bin, text	Assamese: bin, text	Asturian: bin, text
Azerbaijani: bin, text	Bashkir: bin, text	Basque: bin, text
Bavarian: bin, text	Belarusian: bin, text	Bengali: bin, text
Bihari: bin, text	Bishnupriya Manipuri: bin, text	Bosnian: bin, text
Breton: bin, text	Bulgarian: bin, text	Burmese: bin, text
Catalan: bin, text	Cebuano: bin, text	Central Bicolano: bin, text
Chechen: bin, text	Chinese: bin, text	Chuvash: bin, text
Corsican: bin, text	Croatian: bin, text	Czech: bin, text
Danish: bin, text	Divehi: bin, text	Dutch: bin, text
Eastern Punjabi: bin, text	Egyptian Arabic: bin, text	Emilian-Romagnol: bin, text
English: bin, text	Erzya: bin, text	Esperanto: bin, text

Resources

[English word vectors](#)
[Word vectors for 157 languages](#)
[Wiki word vectors](#)
[Aligned word vectors](#)
[Supervised models](#)
[Language identification](#)
[Datasets](#)

1. 'spamFilter.py' 파일 실행

2. "SELECT MODE No.(0 : TRAIN, 1: TEST, 2: INFER) : "에 실행하고자 하는 Mode number 입력

3-0. train mode

4. 프로그램 종료 후, 'models' 폴더 내 생성된 hdf5 파일 확인

3-1. Test mode

4. 사용하고자 하는 모델명 입력(e.g. 11050156_100_0.0512.hdf5)

```
SELECT MODE No.(0 : TRAIN, 1: TEST, 2: INFER) : 1
1
TEST MODEL NAME(.hdf5): 11050156_100_0.0512.hdf5
11050156_100_0.0512.hdf5
```

5. 터미널에 출력되는 loss, accuracy 확인

```
loss: 0.05624603033065796
accuracy : 0.9962499737739563
```

3-2. 2 Infer mode

4. 사용하고자 하는 모델명 입력(e.g. 11050156_100_0.0512.hdf5)

5. 프로그램 종료 후, 'prediction' 폴더 내 생성된 csv 파일 확인

6. 사용 library

- fasttext(0.9.2)

* 설치 가이드¹

Building fasttext python module

In order to build fasttext module for python, use the following:

```
$ git clone https://github.com/facebookresearch/fastText.git
$ cd fastText
$ sudo pip install .
$ # or :
$ sudo python setup.py install
```

- pandas(1.1.1)

¹ fasttext library 설치 가이드, <https://fasttext.cc/docs/en/support.html>

- numpy(1.19.1)
- tensorflow(1.15.0)
- keras(2.3.1)
- scikit-learn(0.23.2)

7. 결론 및 제언

- 스팸 메시지 여부 판별을 위해, binary classification CNN model을 사용하였다. 가장 정확도가 높았던 CNN 모델은 2~6 gram을 사용하고 data augmentation을 사용하지 않은 '11050156_100_0.0512.hdf5' 모델로, 정확도는 약 94%이다.

- augmentation을 적용하지 않은 모델이 augmentation을 적용한 모델보다 정확도가 높는데, 이는 augmented sample data들이 CNN 내 max pooling layer에서는 유사한 모양을 보이기 때문으로 추정된다. 기존 augmentation 방식 대신, padding을 넣은 뒤 truncate 하는 등의 방식을 사용한다면 accuracy가 향상될 가능성이 있다.