**Student ID: 2016025105**

**Name: 강재훈**

목차

1. data 샘플 처리
2. N-gram
3. TFaboutFile
4. TFaboutNtrainset
5. TFaboutMtrainset
6. analysisNtest
7. analysisMtest
8. 실행결과
9. 전체적인 동작 ( IMG파일 )
10. **data 샘플 처리**

텍스트이(가) 표시된 사진

자동 생성된 설명

data 샘플로는 제공해주신 API sequence를 이용하였습니다. 정상적인 API sequence 2250개 중 1번부터 1200번까지의 파일을 이용하였고 비정상적인 API sequence 5250개 중 1번부터 1600번까지의 파일을 이용하였습니다. 이후 python의 train\_test\_split 모듈을 사용하여 정상 파일과 비정상 파일을 train set과 test set으로 나누어서 진행하였습니다. 보통 train set이 test set의 3배정도의 크기를 갖습니다.

1. **N-gram**

텍스트이(가) 표시된 사진

자동 생성된 설명

파일을 열고 한문장씩 받아오며 해당 문장을 map\_key를 이용해 dictionary에 mapping 시켰습니다. 그냥 문장 그대로를 사용하여 N-gram을 진행할 경우 메모리 에러가 발생할 수 있으므로 dictionary를 사용하였습니다. apis에는 파일 전체의 내용이 들어갈 것이고 이를 4개씩 묶는 4-gram을 사용하였습니다. 최종적으로 ret에는 dictionary로 mapping 된 4-gram이 저장됩니다.

1. **TFaboutFile**

텍스트이(가) 표시된 사진

자동 생성된 설명

인자로 받아온 파일 1개에 대해서 4-gram을 실행한 뒤 그 결과를 가지고 서열의 frequency를 셉니다. 역시 python dictionary를 사용하였습니다.

1. **TFaboutNtrainset**

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적인 파일들 중 train set에 대해 나온 서열들을 nTrain\_TF에 저장해줍니다.

1. **TFaboutMtrainset**

텍스트이(가) 표시된 사진

자동 생성된 설명

비정상적인 파일들 중 train set에 대해 서열의 frequency를 측정합니다. 한개의 파일에서 특정 서열이 엄청 많이 나오더라도 +1 만 해줌으로써 특정 파일에 결과가 좌지우지 되는 상황을 방지하였습니다. 또한 TFaboutNtrainset에서 구한 nTrain\_TF를 이용해 일반 파일에서 사용되는 서열들은 비정상적인 파일을 구별하는 서열에서는 제외시켰습니다. 이후 프로그램을 돌려보며 적절한 error\_rate(0.000079) 값을 설정하였고 그 error\_rate보다 큰 빈도수를 가지는 서열들을 최종적으로 mTrain\_TF에 저장해 이 mTrain\_TF를 이용해 이후 정상적인 파일과 비정상적인 파일들을 분류하게 됩니다.

1. **analysisNtest**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

정상적인 파일들에 대해서 test set으로 분류한 파일들을 가지고 정상 파일인지 비정상 파일인지 체크합니다. 파일 각각을 대상으로 출현하는 서열을 구한 뒤 detect\_filter(TFaboutMtrainset에서 구한 mTrain\_TF에 들어있는 서열들)을 이용해서 정상파일인지 비정상 파일인지를 구별합니다.

1. **analysisMtest**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

비정상적인 파일들에 대해서 test set으로 분류한 파일들을 가지고 정상 파일인지 비정상 파일인지 체크합니다. 파일 각각을 대상으로 출현하는 서열을 구한 뒤 detect\_filter(TFaboutMtrainset에서 구한 mTrain\_TF에 들어있는 서열들)을 이용해서 정상파일인지 비정상 파일인지를 구별합니다.

1. **실행결과**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

python의 train\_test\_split 모듈에 따라 정상적인 파일을 train 파일 684개, test 파일 229개로 나눕니다. 비정상적인 파일은 1134개 378개로 나눕니다. 앞서 정상적인 파일을 1200번, 비정상적인 파일을 1600번까지 쓴다고 했는데 차이가 나는 이유는 중간 중간 빠진 번호들이 있고 디렉토리의 파일 리스트를 읽어오는 과정에서 .ds\_store이라는 텍스트 파일과는 무관한 것이 저장되기 때문입니다. 하지만 이후 처리하는 과정에서는 확장자 명이 txt인 파일만 처리하게 코드를 짰으므로 문제가 되지 않습니다.

.

이후 TFaboutNtrainset 함수의 결과로 정상적인 파일들에서 추출한 서열이 총 114642개인것을 확인 할 수 있습니다.

실내, 사진, 앉아있는, 테이블이(가) 표시된 사진

자동 생성된 설명



이후 TFaboutMtrainset함수를 통해 비정상적인 파일들에서 추출한 모든 서열을 tmp\_mTrain에 저장하고 이들중 error\_rate보다 높은 확률을 갖는 서열들을 최종적으로 mTrain\_TF에 저장하였습니다. 총 97개입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이후 추출한 서열들을 가지고 정상적인 파일, 비정상적인 파일들의 train set에 대해 판별해본 결과 입니다. 비정상적인 파일들에서 추출된 서열들 중 정상파 일들에서 추출된 서열들과 중복되는 서열들은 제거 했기 때문에 정상 train set에 있는 파일들은 모두 정상적인 파일로 인식합니다. 비정상적인 파일들 중 정상 파일로 인식하는 경우가 있는 이유는 비정상적인 파일들에서 추출한 모든 서열을 사용하는 것이 아니기 때문에 그렇습니다. error\_rate를 조금 더 낮춘다면 비정상 파일들에서 추출한 더 많은 서열들이 mTrain\_TF에 저장될 것이기 때문에 비정상 파일을 정상 파일로 판단하는 비율을 줄일수 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

test set에 있는 파일들을 판별해본 결과입니다. 여기서 test set의 정상 프로그램을 비정상으로 인식하는 ntom이 5가 나온 이유는 train set에서 비정상 프로그램들이 많이 갖는 서열인데 정상 프로그램이 한번도 가진적이 없었어서 해당 서열을 위험한 서열로 간주하였기 때문입니다. 즉 비정상이나 정상 프로그램 어디에서나 많이 사용되는 서열이고 정상적인 서열이지만 train set의 정상적인 파일들에서 한번도 사용되지 않았던 서열이였기에 위험한 서열로 인식해 해당 서열이 mTrain\_TF에 들어가게 되고 test set의 정상적인 파열일을 만났을때 해당 파일을 비정상으로 인식하게 됩니다. erro\_rate를 조정함으로서 ntom의 값을 줄일 수 있지만 그렇게 하게되면 비정상을 비정상으로 인식하는 mtom 역시 더 줄어들 가능성이 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이 결과는 error\_rate를 0.00009까지 올려 mTrain\_TF에 9개의 서열만이 포함되게 함으로서 생성해낸 결과물입니다. 따라서 사용자의 목적에 따라 error\_rate를 조정할 수 있습니다.

즉 정확도는

error\_rate = 0.000078 일때

test set 총 229개 중 정상을 정상으로 224/229, 비정상을 비정상으로 188/378

error\_rate = 0.00009 일때

test set 총 378개 중 정상을 정상으로 228/229, 비정상을 비정상으로 144/378

입니다. (train set과 test set이 현재와 같이 고정되어 있을 때)

1. **전체적인 동작 ( IMG파일 )**

텍스트, 화이트보드이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명