Documents Taxonomy

LASSO regression을 중심으로

유충현

Updated: 2017/12/17

Overview

1. 들어가기

2. 데이터 전처리

3. 모델 생성

4. 모델 성능 비교

들어가기

서론

Taxonomy는 사전적으로 "사물이나 생명체 등을 분류하기 위해서 사용되는 분류체계"로 해석되며, 분류체계는 트리형의 위계적(Hirerachy) 구조로 표현한다.

Documents Taxonomy는 문서(텍스트)들을 분류체계 기준으로 분류하는 것으로, 대표적인 것에 콜센터의 상담 내용을 상담 분류 체계로 분류하는 것이 있다.

엄밀하게 구분하면 Taxonomy와 Classification은 다른 개념이지만, 여기서는 Classification Model로 Documents Taxonomy의 가능성을 진단해 본다.

문제 제기

"STT(Speech to Text) 기술로 생성한 상담내역 Documents를 Patterns과 Rules 기반이 아닌 Machine Learning 기법으로 분류할 수는 없을까?"

Patterns과 Rules 기반의 분류의 한계점

- 성능의 편차:
  - 작업자의 숙련도에 의한 분류 성능의 편차가 심함.
- 노동 집약적 기술:
  - Patterns을 찾아내는 업무는 지루하고, 끊임없는 탐색과 단순 반복 업무임 .
- 복잡도 증가의 회피:
  - Patterns과 Rules이 복잡도가 증가하면 Rule의 상호 충돌을 회피하기 위한 비용 증가를 동반함.

학습 목표

다음을 네 가지 섹션의 이해를 목표로 학습을 수행한다.

- 데이터 구조의 이해:
  - Vectorization
  - DTM(Document Term Matrix)
- 데이터 처리의 이해:
  - TF-IDF
  - n-gram
  - Feature hashing
- 모델의 이해:
  - LASSO regression
- 패키지의 이해:
  - text2vec
  - glmnet

학습 방법

"사람들의 대화를 들어보면 개인별로 즐겨 사용하는 언어적 특징이 있는 것처럼, 대통령의 연설문에도 개인적 특징이 담겨있지 않을까?"

연설문만으로 어떤 대통령이 연설했는가를 분류

- 데이터:
  - 대통령 연설문 전문
- 분류 방법:
  - 연설문 내용으로 어느 대통령 이름 분류
    - 김대중, 노무현, 이명박
- 성능의 비교:
  - 몇 가지 방법에 대해서 분류의 성능 비교

데이터 준비하기

- 대통령기록연구실 홈페이지
  - http://www.pa.go.kr/research/contents/speech/index.jsp
  - 수집한 데이터를 speech.rda 파일로 제공

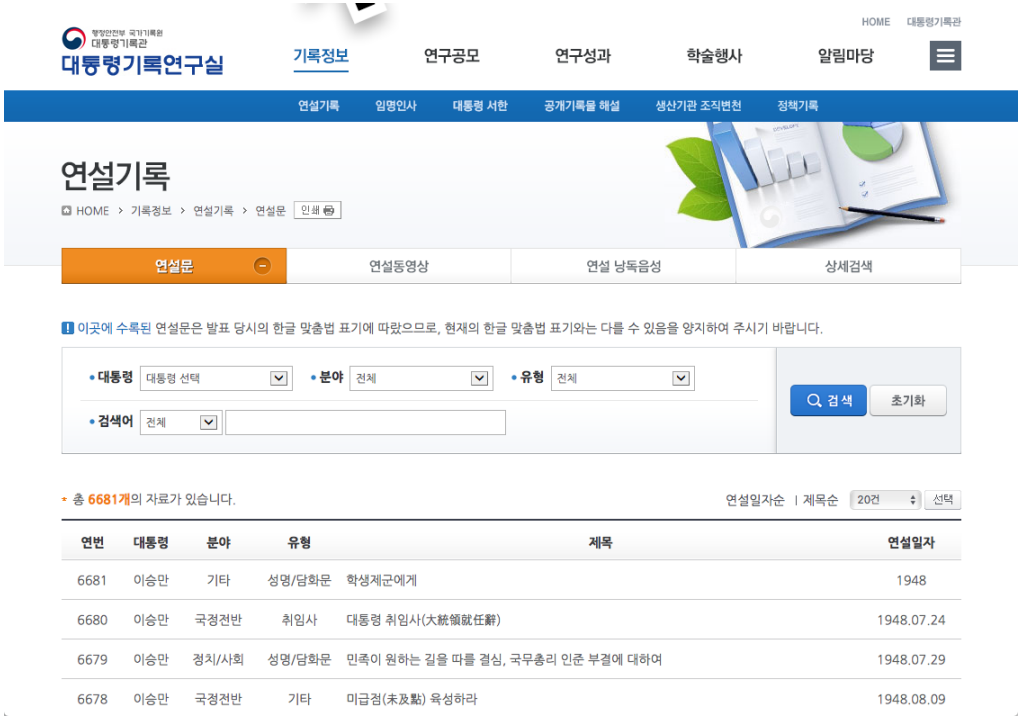


그림: 연설기록 제공 화면

데이터 전처리

데이터 로딩

```
> load("./data/speech.rda")
> dim(speech)


[1] 2408    7


> names(speech)


[1] "id"        "president" "category"  "type"      "title"     "date"
[7] "doc"
```

데이터 구조

```
> table(speech$president)


김대중 노무현 이명박
   822    780    806

> table(speech$category)


 과학기술-정보          교육          국방      국정전반          기타
            75            80           139           308            72
문화-체육-관광      산업-경제      외교-통상      정치-사회          환경
           387           300           678           344            25
```

데이터 구조

```
> table(speech$type)


   국회연설       기념사         기타   기타(연설)  라디오연설       만찬사
         34          490          296           44          100          246
성명-담화문       신년사       취임사          치사       환영사
        223           50            4          907           14

> speech$title[1:2]

[1] "2005 한일 우정의 해 개막식 축사 "
[2] "2007 한중 교류의 해를 맞아 중국 국가주석에게 보내는 축하 메시지"
```

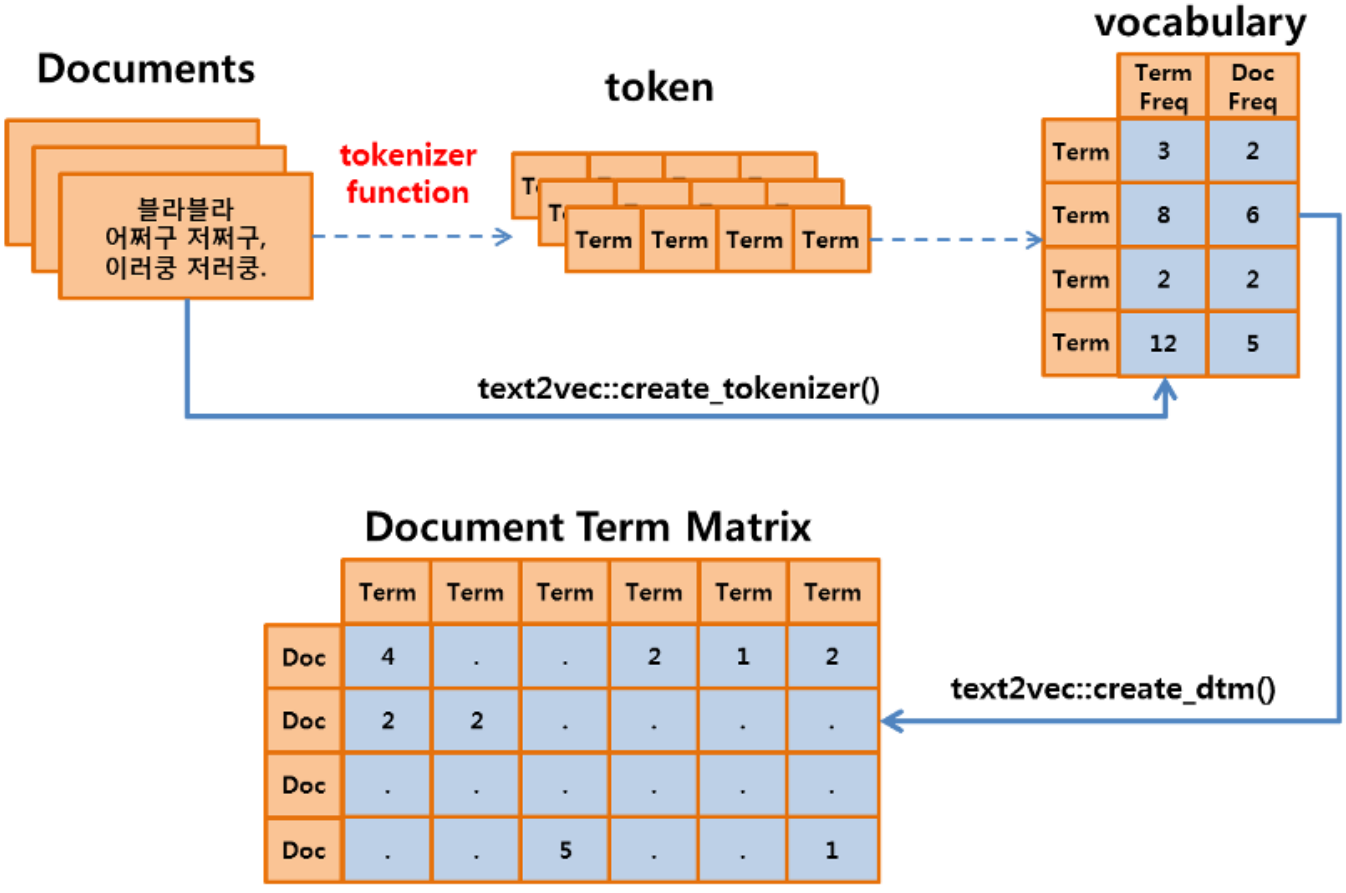
데이터 Split

Train:Test = 7:3로 데이터 Split

```
> library(data.table)
>
> setDT(speech)
> setkey(speech, id)
>
> ids <- speech$id
>
> n <- NROW(speech) * 0.7
>
> set.seed(1234L)
> train_ids <- sample(ids, n)
> test_ids <- setdiff(ids, train_ids)
>
> train <- speech[J(train_ids)]
> test <- speech[J(test_ids)]
```

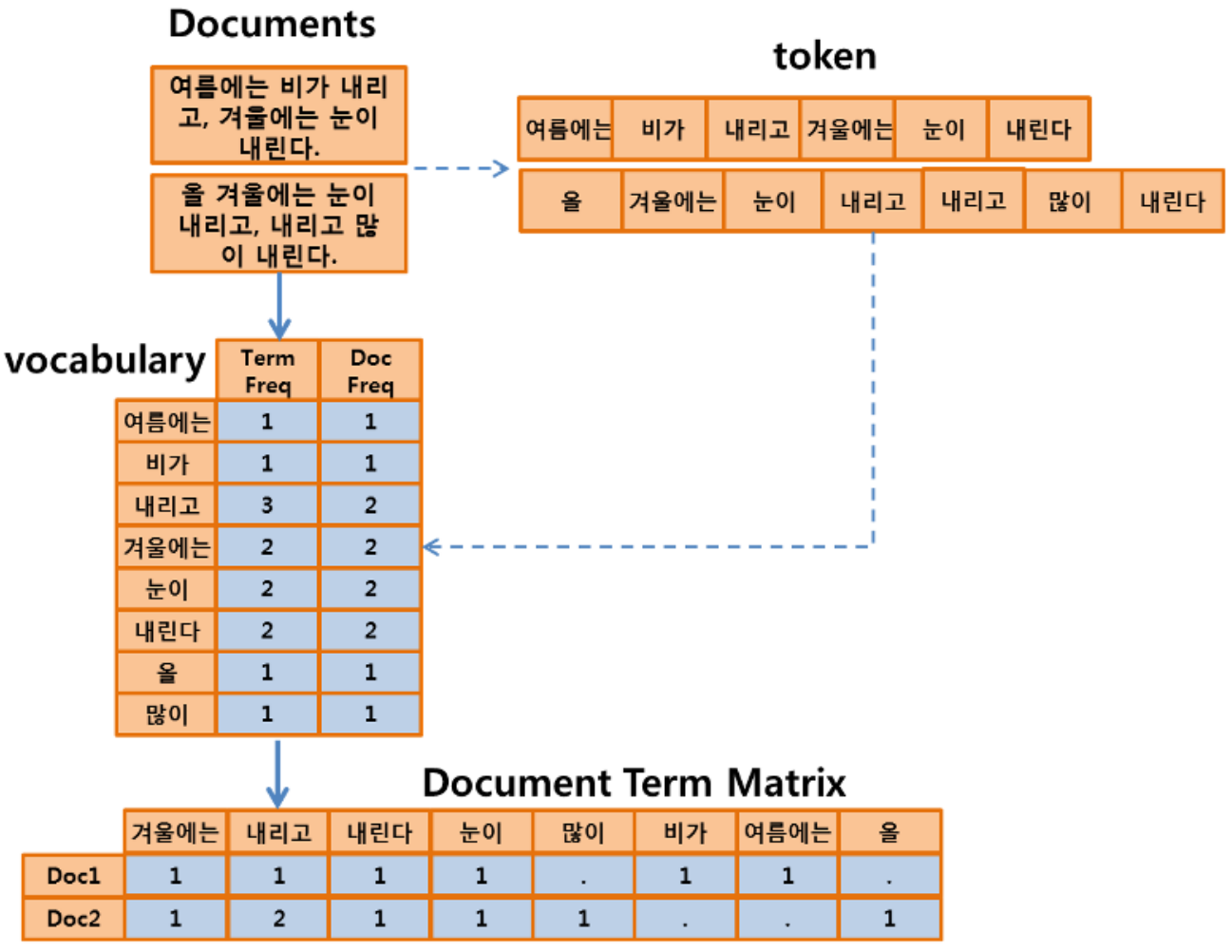
Vectorization : Document Term Matrix 생성

Document Term Matrix 생성 개념



Vectorization : Document Term Matrix 생성

Document Term Matrix 생성 예시



tokenize 반복기 정의

```
> library(text2vec)
>
> token_fun <- tokenizers::tokenize_words
>
> it_train <- itoken_parallel(train$doc,
+                             tokenizer = token_fun,
+                             ids = train$id,
+                             progressbar = FALSE)
>
> it_test <- itoken_parallel(test$doc,
+                            tokenizer = token_fun,
+                            ids = test$id,
+                            progressbar = FALSE)
```

Vocabulary 생성

```
> library(doParallel)
>
> nc <- parallel::detectCores()
> registerDoParallel(nc)
>
> vocab <- create_vocabulary(it_train)
> tail(vocab, n = 3) %>%
+  knitr::kable(., row.names = FALSE, format.args = list(big.mark = ","))
```

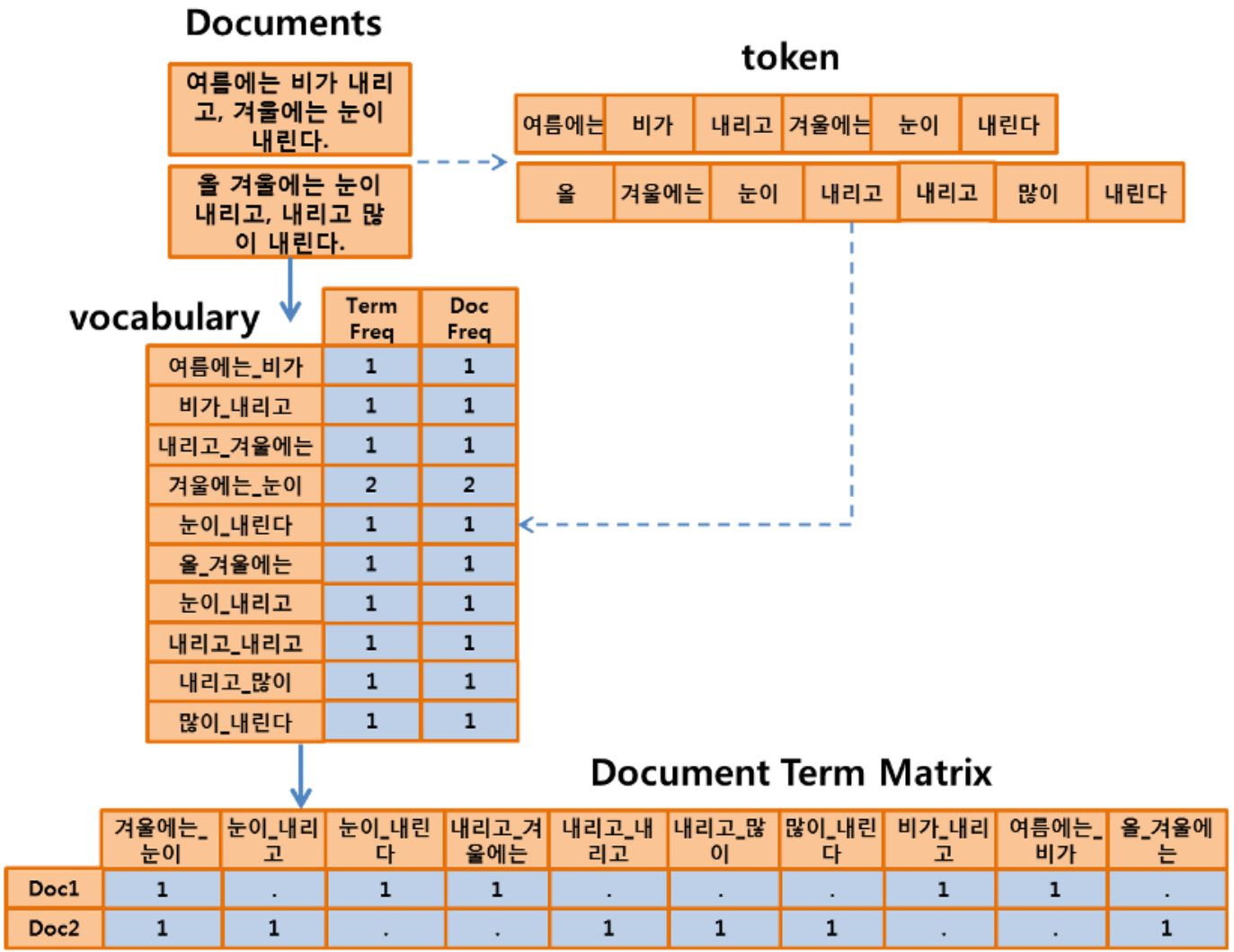
term	term_count	doc_count
수	8,868	1,390
것입니다	9,604	1,448
있습니다	12,409	1,533

Document Term Matrix 생성 : Frequency 기반

```
> vectorizer <-  vocab_vectorizer(vocab)
>
> dtm_train <- create_dtm(it_train, vectorizer)
> dim(dtm_train)


[1]   1685 130282


> dtm_test <- create_dtm(it_test, vectorizer)
> dim(dtm_test)


[1]    723 130282
```

N-Grams 기반 DTM 생성

N-Grams 기반 DTM 생성 예시



N-Grams 기반 DTM 생성

Vocabulary 생성

```
> vocab_bigram <- create_vocabulary(it_train, ngram = c(1L, 2L))
>
> dim(vocab_bigram)


[1] 795447      3


> head(vocab_bigram, n = 3) %>%
+  knitr::kable(., row.names = FALSE, format.args = list(big.mark = ","))
```

term	term_count	doc_count
새로운_원천기술을	1	1
민주주의재단_national	1	1
다차원적으로	1	1

N-Grams 기반 DTM 생성

Document Term Matrix 생성

prune_vocabulary() : vocabulary 가지치기 (처리속도 개선)

```
> vocab_bigram <- vocab_bigram %>%
+   prune_vocabulary(term_count_min = 10, doc_proportion_max = 0.5)
> vectorizer_bigram <- vocab_vectorizer(vocab_bigram)
>
> dtm_train_bigram <- create_dtm(it_train, vectorizer_bigram)
> dim(dtm_train_bigram)


[1]  1685 19558


> dtm_test_bigram  <- create_dtm(it_test, vectorizer_bigram)
> dim(dtm_test_bigram)


[1]   723 19558
```

Feature hashing 기반 DTM 생성

Document Term Matrix 생성

```
> vectorizer_hash <- hash_vectorizer(hash_size = 2 ^ 14, ngram = c(1L, 2L))
>
> dtm_train_hash <- create_dtm(it_train, vectorizer_hash)
> dim(dtm_train_hash)

[1]  1685 16384

> dtm_test_hash <- create_dtm(it_test, vectorizer_hash)
> dim(dtm_test_hash)

[1]   723 16384
```

TF-IDF 기반 DTM 생성

Document Term Matrix 생성

* TF-IDF : Term Frequency - Inverse Document Frequency

"전체 문서들에서 출현빈도는 낮지만, 특정 문서에서 출현빈도가 높은 Term은 특정 문서들을 구별해주는 유용한 Term이다."

```
> tfidf <- TfIdf$new()
> dtm_train_tfidf <- fit_transform(dtm_train, tfidf)
> dim(dtm_train_tfidf)

[1]   1685 130282

> dtm_test_tfidf <- create_dtm(it_test, vectorizer) %>%
+   transform(tfidf)
> dim(dtm_test_tfidf)

[1]    723 130282
```

모델 생성

LASSO(Least Absolute Shrinkage Selector Operator)

정규화 선형회귀모형 (Regularized Regression)

- 관측치의 수 < 독립변수의 수 –> overfiting
  - 회귀계수가 과도하게 증가하는 경향.
- 정규화 선형회귀모형:
  - 추가 제약조건으로 계수의 크기를 제한하는 방법
  - cost 함수를 최소화하는 조건으로 회귀계수 도출
- 정규화 선형회귀모형 종류:
  - Ridge regression
  - LASSO regression
  - Elastic Net regression

cost 함수

정규화 선형회귀모형의 cost 함수와 특징

- Ridge regression
  - $cost = \sum e_i^2 + \lambda \sum w_i^2$
  - 회귀계수를 줄여주는 효과 –> robust
- LASSO refression
  - $cost = \sum e_i^2 + \lambda \sum |w_i|$
  - 회귀계수를 줄여주는 효과 –> robust
  - 유의미하지 않는 변수의 회귀계수 0 –> 변수선택 효과
- Elastic Net regression
  - $cost = \sum e_i^2 + \lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$
  - Ridge와 LASSO의 hybrid

LASSO regression : Frequency 모델 생성

```
> library(glmnet)
>
> NFOLDS <- 10
> classifier <- cv.glmnet(x = dtm_train, y = train[['president']],
+                         family = 'multinomial',
+                         # L1 penalty
+                         alpha = 1,
+                         # interested in the area under ROC curve
+                         type.measure = "auc",
+                         # n-fold cross-validation
+                         nfolds = NFOLDS,
+                 # high value is less accurate, but has faster training
+                         thresh = 1e-3,
+                 # again lower number of iterations for faster training
+                         maxit = 1e3,
+                         parallel = TRUE)
```

LASSO regression - Frequency 모델 평가

```
> library(caret)
>
> pred_voca <- predict(classifier, dtm_test, type = 'response')[, , 1]
> president_voca <- apply(pred_voca, 1,
+                         function(x) colnames(pred_voca)[which(max(x) == x)])
>
> cm_voca <- confusionMatrix(test$president, president_voca)
```

LASSO regression : Frequency 모델 평가

```
> cm_voca$table

          Reference
Prediction 김대중 노무현 이명박
    김대중    214     17      7
    노무현      2    233      8
    이명박      0     12    230

> cm_voca$overall

     Accuracy         Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
  9.363762e-01  9.045312e-01  9.160437e-01  9.530465e-01  3.623790e-01
AccuracyPValue  McnemarPValue
 5.307683e-235   2.013406e-04
```

LASSO regression : Frequency 모델 평가

```
> cm_voca$byClass

                Sensitivity Specificity Pos Pred Value Neg Pred Value
Class: 김대중   0.9907407   0.9526627      0.8991597      0.9958763
Class: 노무현   0.8893130   0.9783080      0.9588477      0.9395833
Class: 이명박   0.9387755   0.9748954      0.9504132      0.9688150
                Precision    Recall        F1 Prevalence Detection Rate
Class: 김대중 0.8991597 0.9907407 0.9427313  0.2987552      0.2959889
Class: 노무현 0.9588477 0.8893130 0.9227723  0.3623790      0.3222683
Class: 이명박 0.9504132 0.9387755 0.9445585  0.3388658      0.3181189
                Detection Prevalence Balanced Accuracy
Class: 김대중           0.3291840         0.9717017
Class: 노무현           0.3360996         0.9338105
Class: 이명박           0.3347165         0.9568355
```

LASSO regression : N-Grams

```
> classifier <- cv.glmnet(x = dtm_train_bigram, y = train[['president']],
+                         family = 'multinomial',
+                         alpha = 1,
+                         type.measure = "auc",
+                         nfolds = NFOLDS,
+                         thresh = 1e-3,
+                         maxit = 1e3)
```

LASSO regression : N-Grams 모델 평가

```
> pred_bigram <- predict(classifier, dtm_test_bigram,
+                        type = 'response')[, , 1]
>
> president_bigram <- apply(pred_bigram, 1,
+                           function(x)
+                          colnames(pred_bigram)[which(max(x) == x)])
>
> cm_bigram <- confusionMatrix(test$president, president_bigram)
```

LASSO regression : N-Grams 모델 평가

```
> cm_bigram$table

          Reference
Prediction 김대중 노무현 이명박
    김대중    223     10      5
    노무현      3    231      9
    이명박      2     15    225

> cm_bigram$overall

     Accuracy         Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
  9.391425e-01  9.086961e-01  9.191596e-01  9.554363e-01  3.540802e-01
AccuracyPValue  McnemarPValue
 2.000079e-244   8.752061e-02
```

LASSO regression : N-Grams 모델 평가

```
> cm_bigram$byClass


                Sensitivity Specificity Pos Pred Value Neg Pred Value
Class: 김대중   0.9780702   0.9696970      0.9369748      0.9896907
Class: 노무현   0.9023438   0.9743041      0.9506173      0.9479167
Class: 이명박   0.9414226   0.9648760      0.9297521      0.9708940
                Precision    Recall        F1 Prevalence Detection Rate
Class: 김대중 0.9369748 0.9780702 0.9570815  0.3153527      0.3084371
Class: 노무현 0.9506173 0.9023438 0.9258517  0.3540802      0.3195021
Class: 이명박 0.9297521 0.9414226 0.9355509  0.3305671      0.3112033
                Detection Prevalence Balanced Accuracy
Class: 김대중           0.3291840         0.9738836
Class: 노무현           0.3360996         0.9383239
Class: 이명박           0.3347165         0.9531493
```

LASSO regression : Feature hashing

```
> classifier <- cv.glmnet(x = dtm_train_hash, y = train[['president']],
+                         family = 'multinomial',
+                         alpha = 1,
+                         type.measure = "auc",
+                         nfolds = NFOLDS,
+                         thresh = 1e-3,
+                         maxit = 1e3,
+                         parallel = TRUE)
```

LASSO regression : Feature hashing 모델 평가

```
> pred_hash <- predict(classifier,
+                      dtm_test_hash, type = 'response')[, , 1]
>
> president_hash <- apply(pred_hash, 1,
+                         function(x)
+                           colnames(pred_hash)[which(max(x) == x)])
>
> cm_hash <- confusionMatrix(test$president, president_hash)
```

LASSO regression : Feature hashing 모델 평가

```
> cm_hash$table

          Reference
Prediction 김대중 노무현 이명박
    김대중    215     18      5
    노무현      6    227     10
    이명박      5     13    224

> cm_hash$overall

     Accuracy         Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
 9.211618e-01  8.817166e-01  8.990602e-01  9.397445e-01  3.568465e-01
AccuracyPValue  McnemarPValue
 2.667352e-224   9.404918e-02
```

LASSO regression : Feature hashing 모델 평가

```
> cm_hash$byClass

                Sensitivity Specificity Pos Pred Value Neg Pred Value
Class: 김대중   0.9513274   0.9537223      0.9033613      0.9773196
Class: 노무현   0.8798450   0.9655914      0.9341564      0.9354167
Class: 이명박   0.9372385   0.9628099      0.9256198      0.9688150
                Precision    Recall        F1 Prevalence Detection Rate
Class: 김대중 0.9033613 0.9513274 0.9267241  0.3125864      0.2973721
Class: 노무현 0.9341564 0.8798450 0.9061876  0.3568465      0.3139696
Class: 이명박 0.9256198 0.9372385 0.9313929  0.3305671      0.3098202
                Detection Prevalence Balanced Accuracy
Class: 김대중           0.3291840         0.9525249
Class: 노무현           0.3360996         0.9227182
Class: 이명박           0.3347165         0.9500242
```

LASSO regression : TF-IDF

```
> classifier <- cv.glmnet(x = dtm_train_tfidf, y = train[['president']],
+                         family = 'multinomial',
+                         alpha = 1,
+                         type.measure = "auc",
+                         nfolds = NFOLDS,
+                         thresh = 1e-3,
+                         maxit = 1e3,
+                         parallel = TRUE)
```

LASSO regression : Feature hashing 모델 평가

```
> pred_tfidf <- predict(classifier,
+                       dtm_test_tfidf, type = 'response')[, , 1]
>
> president_tfidf <- apply(pred_tfidf, 1,
+                          function(x)
+                           colnames(pred_tfidf)[which(max(x) == x)])
>
> cm_tfidf <- confusionMatrix(test$president, president_tfidf)
```

LASSO regression : TF-IDF 모델 평가

```
> cm_tfidf$table

          Reference
Prediction 김대중 노무현 이명박
    김대중    230      5      3
    노무현      7    226     10
    이명박      4      6    232

> cm_tfidf$overall

     Accuracy         Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
 9.515906e-01  9.273867e-01  9.333144e-01  9.660528e-01  3.388658e-01
AccuracyPValue  McnemarPValue
 1.183254e-270   6.877764e-01
```

LASSO regression : TF-IDF 모델 평가

```
> cm_tfidf$byClass


                Sensitivity Specificity Pos Pred Value Neg Pred Value
Class: 김대중   0.9543568   0.9834025      0.9663866      0.9773196
Class: 노무현   0.9535865   0.9650206      0.9300412      0.9770833
Class: 이명박   0.9469388   0.9790795      0.9586777      0.9729730
                Precision    Recall        F1 Prevalence Detection Rate
Class: 김대중 0.9663866 0.9543568 0.9603340  0.3333333      0.3181189
Class: 노무현 0.9300412 0.9535865 0.9416667  0.3278008      0.3125864
Class: 이명박 0.9586777 0.9469388 0.9527721  0.3388658      0.3208852
                Detection Prevalence Balanced Accuracy
Class: 김대중           0.3291840         0.9688797
Class: 노무현           0.3360996         0.9593035
Class: 이명박           0.3347165         0.9630091
```

모델 성능 비교

모델 성능 비교

```
> result <- rbind(cm_voca$overall, cm_bigram$overall,
+                 cm_hash$overall, cm_tfidf$overall)
> data.frame(method=c("Frequency", "Bigram", "Hash", "TF-ID"),
+            round(result, 4)) %>%
+   dplyr::select(method, Accuracy, AccuracyLower, AccuracyUpper) %>%
+   dplyr::arrange(desc(Accuracy)) %>%
+   knitr::kable()
```

method	Accuracy	AccuracyLower	AccuracyUpper
TF-ID	0.9516	0.9333	0.9661
Bigram	0.9391	0.9192	0.9554
Frequency	0.9364	0.9160	0.9530
Hash	0.9212	0.8991	0.9397

어떻게 튜닝할 것인가?

"예전에는 성능 튜닝을 모델 튜닝의 협의적인 방법만 생각했었다!"

Text Analytics는 모델과의 싸움이 아니야

Text Analytics에서 예측의 성능을 높히기 위해서는 Natural Language Processing과 Documents를 Vectorization하는 방법을 고민해야 함.

데이터 전처리의 노고는 결과를 배신하지 않는다.

Document Taxonomy를 위한 제언

- tokenizer
  - 형태소 단위의 tokenizer가 words 단위의 tokenizer보다 성능이 낮았다
  - 형태소 단위의 분석만 고집하지 말자.
- trade off
  - DTM은 많은 컬럼을 포함한 대용량 데이터로 연산 속도의 저하 유발
  - 성능 감소를 감내하고, 차원을 축소해 보자.
    - 형태소 단위, pruned matrix, Feature hashing
- hybrid 접근
  - N-Gram DTM을 TF-IDF로 변환하면 어떨까?
  - Accuracy = 0.9627로 TF-IDF보다 높게 나옴
- 전처리
  - 본 예제에서 Text 전처리를 수행하지 않았음

hybrid 접근 코드

```
> tfidf <- TfIdf$new()
> dtm_train_tfidf2 <- fit_transform(dtm_train_bigram, tfidf)
> dtm_test_tfidf2 <- fit_transform(dtm_test_bigram, tfidf)
>
> classifier <- cv.glmnet(x = dtm_train_tfidf2, y = train[['president']],
+                         family = 'multinomial', alpha = 1,
+                         type.measure = "auc", nfolds = NFOLDS,
+                        thresh = 1e-3, maxit = 1e3, parallel = TRUE)
>
> pred_tfidf2 <- predict(classifier, dtm_test_tfidf2,
+                        type = 'response')[, , 1]
> president_tfidf2 <- apply(pred_tfidf2, 1,
+                           function(x)
+                           colnames(pred_tfidf2)[which(max(x) == x)])
> cm_tfidf2 <- confusionMatrix(test$president, president_tfidf2)
```

References

Vectorization

http://text2vec.org/vectorization.html

Dmitriy Selivanov, 2017

Lasso and Elastic-Net Regularized Generalized Linear Models

https://cran.r-project.org/web/packages/glmnet/vignettes/glmnet_beta.pdf

Trevor Hastie and Junyang Qian, 2016

THE END