Dmitry Kutsev, Comparation of the ebbedding models for sentiment analysis

1.1 Data description.

Dataset consists of grouped verbs, which were manually annotated by the group of 3 people into 2 sets: positive and neganive. Thees verbs were also annotated by another group of 5 people into 2 sets: absence or presence of the tonality. The “polar_or_not” column in the dataset shows the sum of people who marked the verb as tonal. The verbs, which have 1 or 0 tonality votes were removed from the dataframe.

One of the aims of this research is to check if there is any connection between number of people, who annotated verbs as tonal, and positiveannotations. Our hypotesys based on the assumption, that negative verbs have stronger polarity, than positive ones do.

a0.a1_pol column of the dataframe includes annotations of agens to patiens relations of the verb arguments handled by the RuSentiFrames lexicon experts. We can use epy verb “использовать”(to use) as an example: “a0” - the one, who uses(or the ones who use). “a1” - something or someone being used.

Cosine distance, Cosine distance 2 and 3 columns includes the results of the work of classification models, built with SemanticAxis method(details in the pragraph below). Fasttext Scipgram and Fasttext CBOW word embedding models were trained on different corpuses. We are gratefull to RusVectores web resource for the models we took from it. My next hypotesys examines if there is statistically significant connection between cosine distance values of different models. This will be checked with Spearman correlation test. After that, I build a logistic regression classification model with cosine distances of one model as single predictors, and with cosine distances of several models, as multiple predictors.

1.2 Previous research.

To proceed with experiments on a word-based level, from all the predicates present in the lexicon we selected the most frequent ones, building a list of 1000 verbs on news corpora. We naturally had to spare frame entries representing predicates followed by prepositions, as well as idioms and other multiword expressions. We also spared nominal predicates to maintain consistency of the dataset. Thus, at the initial stage we focused our research on the 2794 verbal predicates expressed by a single word.

With Python scripts we intersected the most frequent verbs from the list of 2794 predicates with the most frequent verbs from the Lenta news corpus. First 1000 intersections were chosen and put into a list for further processing. Ира - про классы Bearing in mind the complex structure of the lexicon we attempted further subdivision of frame entries to classes on the basis of negative or positive direction of sentiment attitudes associated with the entries ("polarity" slot). The approach we took involved reshaping the lexicon and taking advantage of the way tables are handled by pandas Python library. This allowed us to group the verbs into several intersecting classes. Initially we grouped verbs having at least one pair of arguments expressing same attitudes towards each other into 6 categories. For example, verbs in the category 'pos_a0_a1_mutual' are part of the frames characterised by mutually positive attitudes of the alleged agent and patient. Though the lexicon creators claim that mainly these notations ('a0', 'a1', etc.) correctly indicate most frequent roles throughout the lexicon, having explored the data we observed cases of inconsistency that included, for example, both animate and inanimate participants under the same argument notation.

We took several steps to detect predicates that address the needs of our experimental set and are thought to bear unambiguous sentiment. The resulting table includes 445 single word predicates marked with 'INFN' or 'VERB' pos-tag by pymorphy2 Python library. 322 verbs were manually annotated negative by all 3 annotators and 123 verbs were annotated with positive labels. For each verb there is also information about the number of annotators (from 1 to 6) that considered the verb as bearing sentiment as opposed to the verbs dropped after having been labelled as bearing sentiment by 0 out of 6 annotators. Other columns include case tags of two arguments in Open Corpora notation. The case tags have been defined manually by our research team with regard to future experiments on the sentence level, when we'll be working with arguments put in corresponding case forms. For the same purpose some verbs were marked as one-participant in the commentary column. In further experiments, we chose semantic seed sets, and calculated semantic axis according to Semantic Axis Method(Jurafsky & Martin, 2019). We also used models(Word2Vec, Fasttext mostly) from the RusVectōrēs web resource to measure cosine distances between verbs from RuSentiFrames lexicon and the semantic axis.

We used fasttext models to vectorize verbs within seed sets:

positive_multi_seed = ['одобрять', 'хвалить', 'поощрять', 'любить']
negative_multi_seed = ['ненавидеть', 'убить', 'ругать', 'злиться']

Models were used:

Model1 - Fasttext skip gram model, trained он Tayga dataset:

Model2 - Fasttext CBOW model, trained он Araneum dataset:

Model3, Fasttext scipgram, trained on National Russian corpus dataset:

This is what the dataset looks like:

```
senti_df <-
read.csv("https://raw.githubusercontent.com/DmitryKutsev/NIS_SentiFrame
/master/my_senti_df2.csv")
senti_df

##                    verb polar_or_not polarity_summ
## 1          арестовывать            4             0
## 2             атаковать            4             0
## 3                беречь            3             3
## 4            беспокоить            2             0
## 5                  бить            3             0
## 6            благодарить           4             3
## 7            блокировать           4             0
## 8                бомбить            3             0
## 9               бороться            4             0
## 10               бояться            3             0

##      падеж.второго.аргумента.в.нотации.pymorphy2 a0.a1_pol
fasttext_polarity
## 1                                               accs         0
0
## 2                                               accs         0
0
## 3                                               accs         1
1
## 4                                               accs         0
0
## 5                                               accs         0
0
## 6                                               accs         1
1
## 7                                               accs         0
0
## 8                                               accs         0
0
## 9                                               ablt         0
0
## 10                                              gent         0
accs         1                  1
##     manual_polarity cosine_distance cosine_distance2
fasttext_polarity2
## 1                0    -0.047116164    -0.1625133455
0
## 2                0    -0.170026302    -0.1094973385
```

```
0
## 3                 1     0.055182472    -0.0226197112
0
## 4                 0    -0.252901465    -0.1226359308
0
## 5                 0    -0.356272012    -0.2701320350
0
## 6                 1     0.257343948     0.1845448762
1
## 7                 0    -0.027648978    -0.0259298012
0
## 8                 0    -0.297227144    -0.2703138888
0
## 9                 0    -0.215196982    -0.1019719392
0
## 10                0    -0.358463883    -0.1732861698
0
##     cosine_distance3 fasttext_polarity3
## 1       -0.156714648                  0
## 2       -0.305128217                  0
## 3       -0.137535647                  0
## 4       -0.336846292                  0
## 5       -0.474794090                  0
## 6        0.180103078                  1
## 7       -0.086124860                  0
## 8       -0.494721830                  0
## 9       -0.205744222                  0
## 10      -0.408281624                  0
```

2.1 Data preparetion

I decided to drop unnecessary columns and rename polar_or_not column to "polar_votes":

```
senti_df <- subset(senti_df, select = c(verb, polar_or_not,
manual_polarity, cosine_distance, cosine_distance2, cosine_distance3))

senti_df <- senti_df %>%
  rename(
    polar_votes = polar_or_not
    )
senti_df
```

```
##                  verb polar_votes manual_polarity cosine_distance
## 1        арестовывать           4               0    -0.047116164
## 2           атаковать           4               0    -0.170026302
## 3              беречь           3               1     0.055182472
```

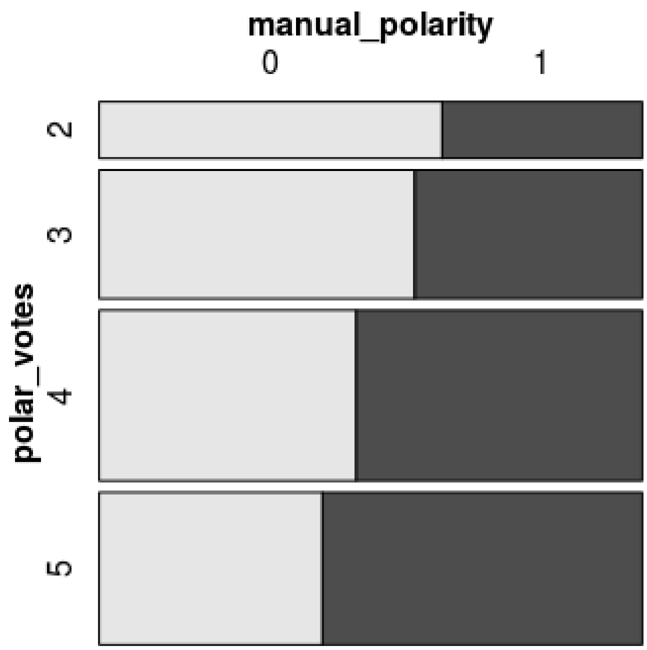
```
## 4          беспокоить          2              0    -0.252901465
## 5                бить          3              0    -0.356272012
## 6          благодарить          4              1     0.257343948
## 7          блокировать          4              0    -0.027648978
## 8              бомбить          3              0    -0.297227144
## 9             бороться          4              0    -0.215196982
## 10             бояться          3              0    -



##     cosine_distance2 cosine_distance3
## 1      -0.1625133455     -0.156714648
## 2      -0.1094973385     -0.305128217
## 3      -0.0226197112     -0.137535647
## 4      -0.1226359308     -0.336846292
## 5      -0.2701320350     -0.474794090
## 6       0.1845448762      0.180103078
## 7      -0.0259298012     -0.086124860
## 8      -0.2703138888     -0.494721830
## 9      -0.1019719392     -0.205744222
## 10     -0.1732861698     -0.408281624
```

Because of imbalanced proportion of positive and negative verbs, let us sort out equal subsets to rut chisquare test.

```
manual_pos <- subset(senti_df, manual_polarity == 1)
manual_neg <- subset(senti_df, manual_polarity == 0)
manual_neg <- slice(manual_neg, 1:nrow(manual_pos))
nrow(manual_pos)

## [1] 85

nrow(manual_neg["manual_polarity"])

## [1] 85

equal_df <- rbind(manual_pos, manual_neg)
                                   3. Experiments.


3.1 Chisquare test
```

Lets assume that the coefficient of tonality is the number of people who marked the verb as tonal.Now we can check if there any connection between manually annotated polarity of the verbs and this coefficient. Proportion can be seen in the mosaic plot below.

```
mosaic(data = equal_df, manual_polarity ~ polar_votes)
```



We can see that amount of positive verbs encreases together with polar votes. This possible to check if there is a connection between polarity votes and positivetonality, using Chisquare test.

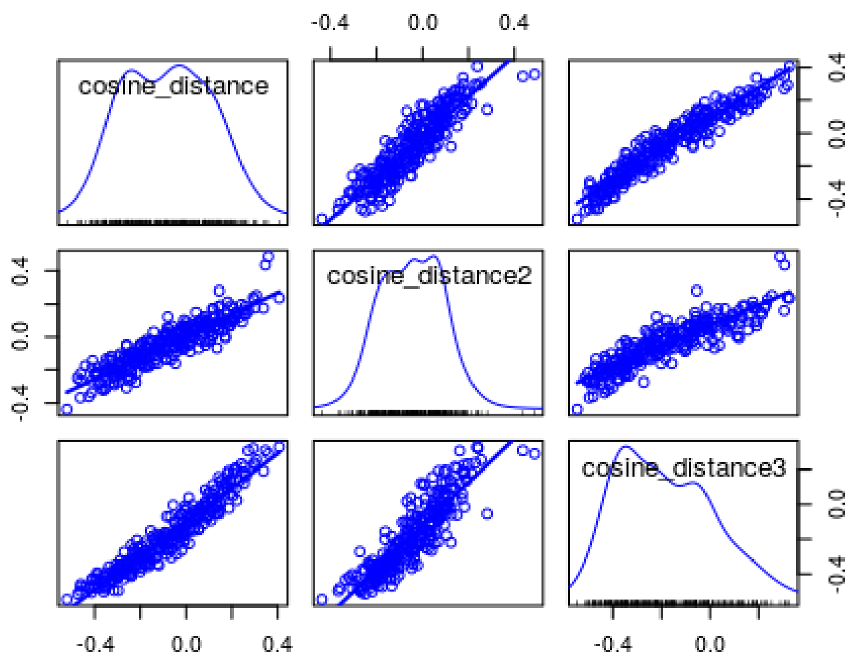
```
table(field=equal_df$manual_polarity , field=equal_df$polar_votes)

##       field
## field  2  3  4  5
##     0 12 25 27 21
##     1  7 18 30 30

chisq.test(equal_df$polar_votes, equal_df$manual_polarity)

##
##  Pearson's Chi-squared test
##
## data:  equal_df$polar_votes and equal_df$manual_polarity
## X-squared = 4.2015, df = 3, p-value = 0.2405
```

P-value > 0.05, do we do not have enough reasons to reject the null-hypothesis. Manual polarity and tonality votes are independent.

```
3.2 Correlation tests
```

Before we start fitting logistic regression let us have a look at the cosine distances of the models 1,2, and 3 with scatterplot matrix.

```
scatterplotMatrix(senti_df[4:6], diagonal = "histogram", smooth =
FALSE)

## Warning in applyDefaults(diagonal, defaults = list(method =
## "adaptiveDensity"), : unnamed diag arguments, will be ignored
```



Besides some outliers, we can see a rather strong connection between cosine distances of all three models. Spearman's correlation test can help to check it formally.

```
cor.test(senti_df$cosine_distance, senti_df$cosine_distance2, method =
"spearman" )

##
##  Spearman's rank correlation rho
##
## data:  senti_df$cosine_distance and senti_df$cosine_distance2
## S = 332668, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.9100401
```

```r
cor.test(senti_df$cosine_distance, senti_df$cosine_distance3, method =
"spearman" )
```

```
##
##  Spearman's rank correlation rho
##
## data:  senti_df$cosine_distance and senti_df$cosine_distance3
## S = 136052, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.9632089
```

```r
cor.test(senti_df$cosine_distance2, senti_df$cosine_distance3, method =
"spearman" )
```

```
##
##  Spearman's rank correlation rho
##
## data:  senti_df$cosine_distance2 and senti_df$cosine_distance3
## S = 359538, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##      rho
## 0.902774
```

This means, that our assumption has been confirmed. Spearman's tests show a srong connection between all three models. The connection between model1 and model3(according to rho coefficient) is stronger than those between the others, which is not surprising, given the same architecture of this models.

To make the analisys more managable it may be useful to transform manual polarity variables from integer to factor.

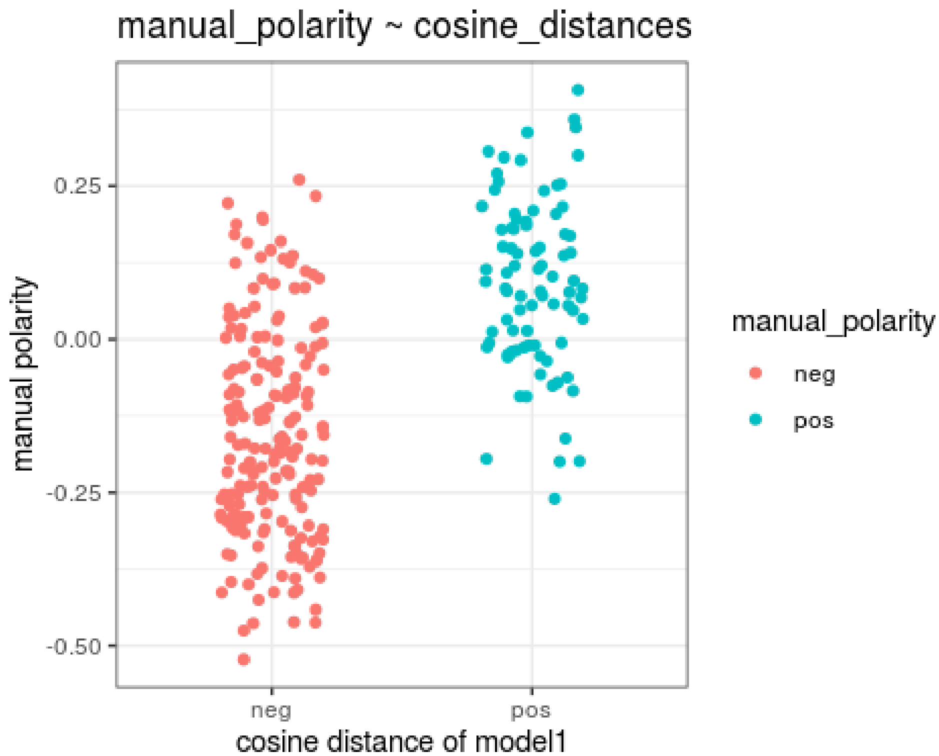
```r
senti_df$manual_polarity <- as.factor(senti_df$manual_polarity)
levels(senti_df$manual_polarity ) <- c("neg", "pos")
senti_df
```

```
##                    verb polar_votes manual_polarity cosine_distance
## 1          арестовывать           4             neg    -0.047116164
## 2             атаковать           4             neg    -0.170026302
## 3                беречь           3             pos     0.055182472
## 4            беспокоить           2             neg    -0.252901465
## 5                  бить           3             neg    -0.356272012
## 6           благодарить           4             pos     0.257343948
## 7           блокировать           4             neg    -0.027648978
## 8               бомбить           3             neg    -0.297227144
## 9              бороться           4             neg    -0.215196982
## 10              бояться           3             neg    -0.358463883
```

```
##     cosine_distance2 cosine_distance3
## 1      -0.1625133455     -0.156714648
## 2      -0.1094973385     -0.305128217
## 3      -0.0226197112     -0.137535647
## 4      -0.1226359308     -0.336846292
## 5      -0.2701320350     -0.474794090
## 6       0.1845448762      0.180103078
## 7      -0.0259298012     -0.086124860
## 8      -0.2703138888     -0.494721830
## 9      -0.1019719392     -0.205744222
## 10     -0.1732861698     -0.408281624
```

We can also check, how negative and positive verbs are distributed between cosine distances. Lets take model1.

```
senti_df %>%
  ggplot(aes(manual_polarity, cosine_distance, color =
manual_polarity))+
  geom_jitter(width = 0.2)+
  labs(title = "manual_polarity ~ cosine_distances",
       x = "cosine distance of model1",
       y = "manual polarity")+
  theme_bw()
```




```
3.3 Logistic regression models
```

To study binomial values I use logistic regression model. To examine both accuracy and AIC values of the logistic regressions, I split data on the train and the test sets.

```
smp_siz = floor(0.7*nrow(senti_df))  # creates a value for dividing the
data into train and test. In this case the value is defined as 75% of
the number of rows in the dataset
smp_siz  # shows the value of the sample size

## [1] 196

set.seed(123)   # set seed to ensure you always have same random
numbers generated
train_ind = sample(seq_len(nrow(senti_df)), size = smp_siz)  # Randomly
identifies therows equal to sample size ( defined in previous
instruction) from  all the rows of Smarket dataset and stores the row
number in train_ind
train = senti_df[train_ind,] #creates the training dataset with row
numbers stored in train_ind
test = senti_df[-train_ind,]  # creates the test dataset excluding the
row numbers mentioned in train_ind
```

I start with cosine distances of the first semantic axis model to fit the model with one predictor.

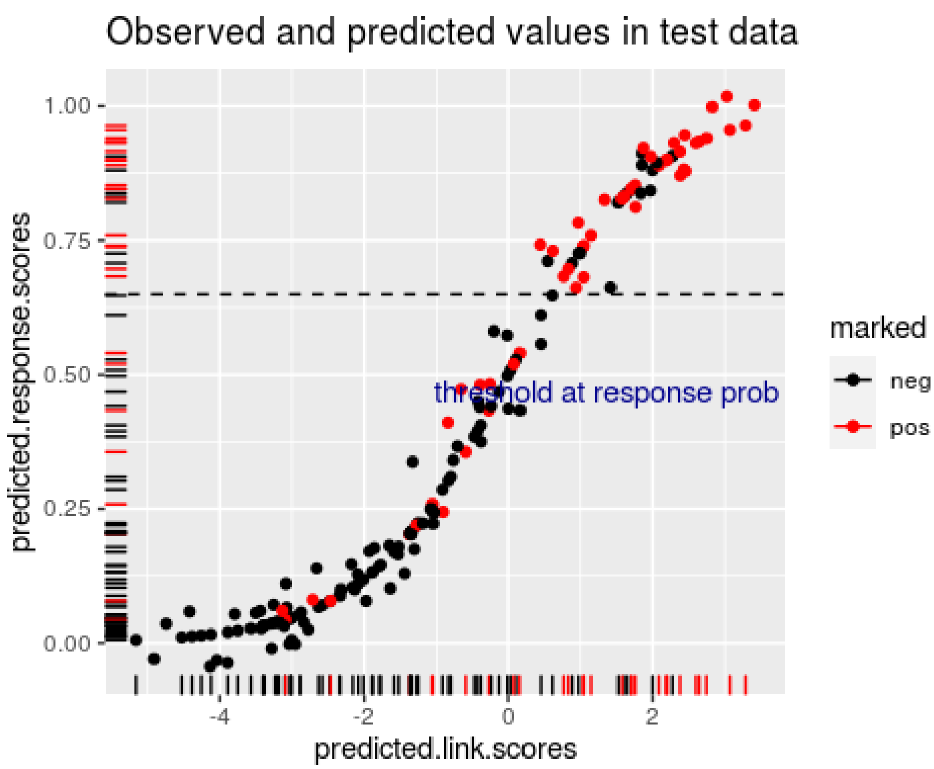
```
fit1 <- glm(manual_polarity~cosine_distance, data = train, family =
"binomial")
fit1_full <- glm(manual_polarity~cosine_distance, data = senti_df,
family = "binomial")
summary(fit1_full)

##
## Call:
## glm(formula = manual_polarity ~ cosine_distance, family =
"binomial",
##     data = senti_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9862  -0.6448  -0.2940   0.6571   2.4651
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.5834     0.1632  -3.574 0.000351 ***
## cosine_distance     9.2456     1.1625   7.953 1.81e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.48  on 280  degrees of freedom
## Residual deviance: 233.84  on 279  degrees of freedom
## AIC: 237.84
##
## Number of Fisher Scoring iterations: 5
```

This results allow us to note down Akaike information criterion(AIC) value for the further comparison. We can also consider the influence of cosine distance values rather strong, according to Standart Estimate value(10.3073).

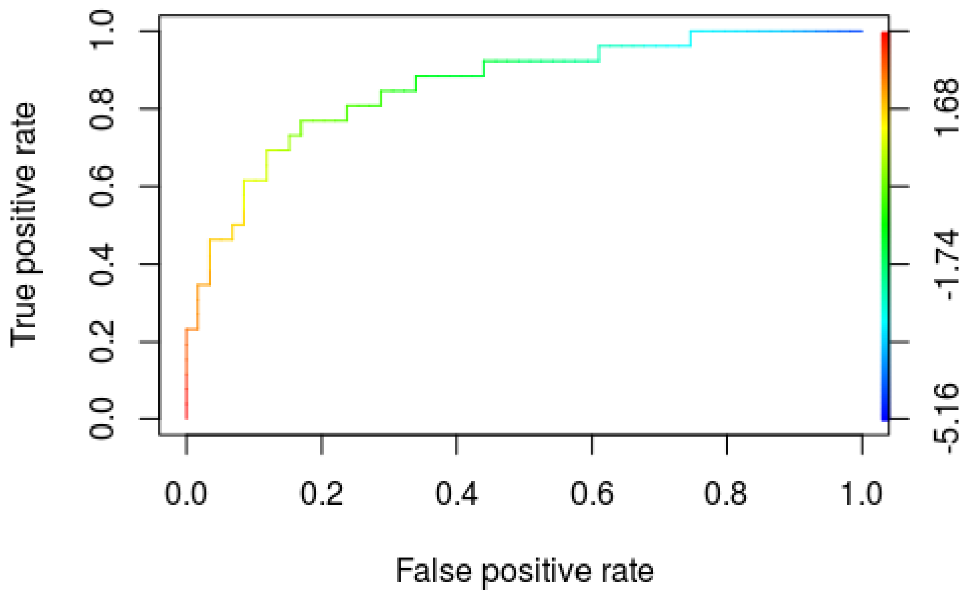
The next step is to check the model trained on the split train data and then predict the values of the linked and the response scores and visualize them.

```
predicted.link.scores <- predict(fit1, newdata=test, type="link")
predicted.response.scores <- predict(fit1, newdata=test,
type="response")
predicted.scores <- data.frame(link=predicted.link.scores,
                         response=predicted.response.scores,
marked=test$manual_polarity,
                         #construction_obs=load.test$CONSTRUCTION,
                         stringsAsFactors=FALSE)
```

```
predicted.scores %>%
  ggplot(aes(x=predicted.link.scores , y=predicted.response.scores ,
col=marked )) +
  scale_color_manual(values=c("black", "red")) +
  geom_point() +
  geom_rug() +
  geom_jitter(width=.5, height=.065) +
  geom_hline(yintercept=0.65, linetype="dashed") +
  annotate(geom="text", x=2, y=0.47, label="threshold at response prob
= 0.65", color="darkblue") +
  ggtitle("Observed and predicted values in test data")
```




```
pred<- predict(fit1, newdata=test)
predicted <- prediction(pred, test$manual_polarity)
predicted <- performance(predicted, "tpr", "fpr")
plot(predicted, colorize=T)
```



Using these graphs we can visualize the prediction rate of our model and choose the prediction rate according to them. The table below shows confusion matrix and statistics which takes account of prediction coefficient 0.65.

```
v <- rep(NA, nrow(predicted.scores))
v <- ifelse(predicted.scores$response >= .65, "pos", "neg")
predicted.scores$tonality_pred <- as.factor(v)
cnf <- confusionMatrix(data = predicted.scores$tonality_pred, reference
= predicted.scores$marked)
cnf

## Confusion Matrix and Statistics
##
##             Reference
## Prediction neg pos
##        neg  52   8
##        pos   7  18
##
##                Accuracy : 0.8235
##                  95% CI : (0.7257, 0.8977)
##     No Information Rate : 0.6941
##     P-Value [Acc > NIR] : 0.005003
##
##                   Kappa : 0.5799
##
```

```
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8814
##             Specificity : 0.6923
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 0.7200
##              Prevalence : 0.6941
##          Detection Rate : 0.6118
##    Detection Prevalence : 0.7059
##       Balanced Accuracy : 0.7868
##
##        'Positive' Class : neg
##
```

The accuracy value is 0.8235. Now we can fit the second regression with cosine distance1 and 2 values as predictors, and compare output metrics.

```
fit12 <- glm(manual_polarity~cosine_distance + cosine_distance2 , data
= train, family = "binomial")
fit12_full <- glm(manual_polarity~cosine_distance + cosine_distance2 ,
data = senti_df, family = "binomial")
summary(fit12_full)

##
## Call:
## glm(formula = manual_polarity ~ cosine_distance + cosine_distance2,
##     family = "binomial", data = senti_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9056  -0.6255  -0.2403   0.5397   2.5165
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.6850     0.1726  -3.969 7.22e-05 ***
## cosine_distance    3.5897     1.9943   1.800  0.07187 .
## cosine_distance2   9.9648     3.0438   3.274  0.00106 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.48  on 280  degrees of freedom
## Residual deviance: 221.93  on 278  degrees of freedom
## AIC: 227.93
##
## Number of Fisher Scoring iterations: 5
```

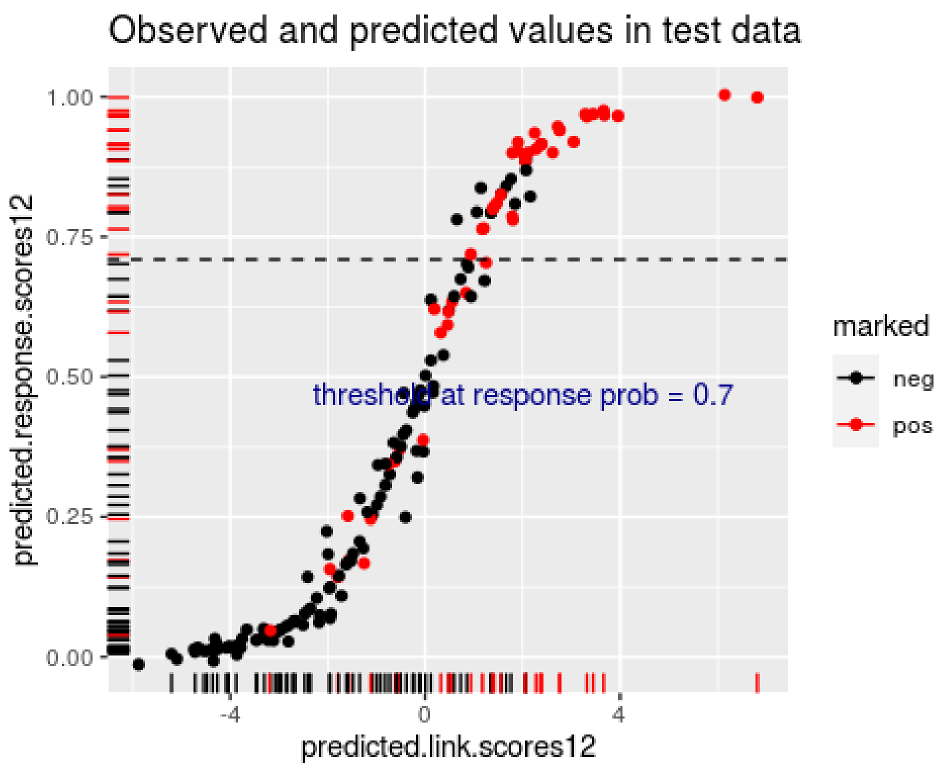
This model shows better AIC value (227.93) than previous one(237.84). Although linear correlation of the cosine distances 1 and 2 showed high value, we can see, that cosine_distance2 values have stronger influence on predictions(9.9648), than cosine_distance1(3.5897).

The next step is to predict scores of this model.

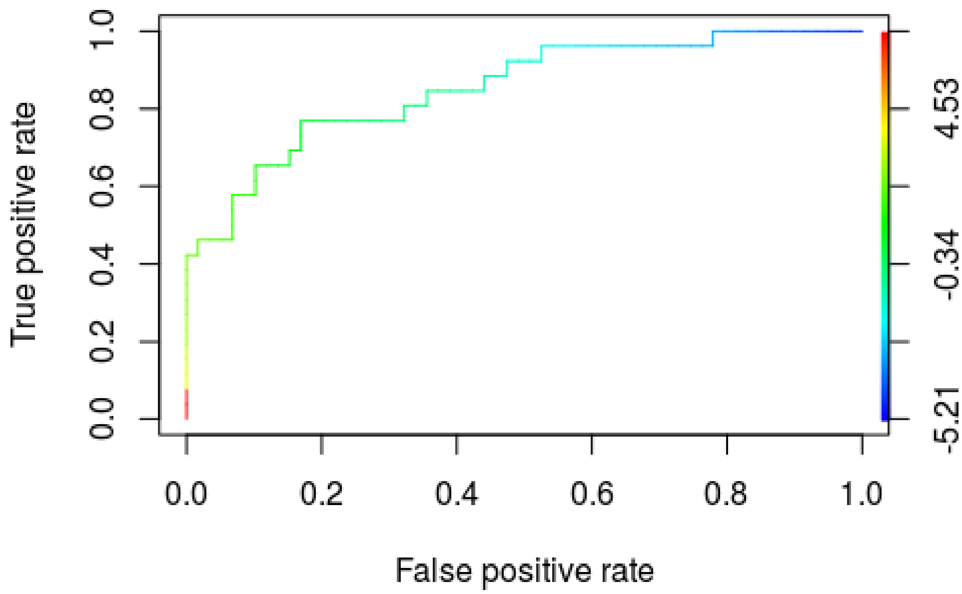
```
predicted.link.scores12 <- predict(fit12, newdata=test, type="link")
predicted.response.scores12 <- predict(fit12, newdata=test,
type="response")
predicted.scores12 <- data.frame(link=predicted.link.scores12,
                                 response=predicted.response.scores12,
marked=test$manual_polarity,
                                 #construction_obs=load.test$CONSTRUCTION,
                                 stringsAsFactors=FALSE)
```

And visualize results.

```
predicted.scores12 %>%
  ggplot(aes(x=predicted.link.scores12 , y=predicted.response.scores12
, col=marked )) +
  scale_color_manual(values=c("black", "red")) +
  geom_point() +
  geom_rug() +
  geom_jitter(width=.70, height=.02) +
  geom_hline(yintercept=0.71, linetype="dashed") +
  annotate(geom="text", x=2, y=0.47, label="threshold at response prob
= 0.7", color="darkblue") +
  ggtitle("Observed and predicted values in test data")
```




```
pred12 <- predict(fit12, newdata=test)
predicted12 <- prediction(pred12, test$manual_polarity)
predicted12 <- performance(predicted12, "tpr", "fpr")
plot(predicted12, colorize=T)
```



Using these graphs we can visualize the prediction rate of our model and choose the prediction rate according to them. The table below shows confusion matrix and statistics which takes account of prediction coefficient 0.7.

```
v <- rep(NA, nrow(predicted.scores12))
v <- ifelse(predicted.scores12$response >= .7, "pos", "neg")
predicted.scores12$tonality_pred <- as.factor(v)
#predicted.scores$tonality_pred
#predicted.scores$marked
cnf12 <- confusionMatrix(data = predicted.scores12$tonality_pred,
reference = predicted.scores$marked)
cnf12

## Confusion Matrix and Statistics
##
##             Reference
## Prediction neg pos
##        neg  52   9
##        pos   7  17
##
##                Accuracy : 0.8118
##                  95% CI : (0.7124, 0.8884)
##     No Information Rate : 0.6941
##     P-Value [Acc > NIR] : 0.01029
##
```

```
##                  Kappa : 0.547
##
##  Mcnemar's Test P-Value : 0.80259
##
##            Sensitivity : 0.8814
##            Specificity : 0.6538
##         Pos Pred Value : 0.8525
##         Neg Pred Value : 0.7083
##             Prevalence : 0.6941
##         Detection Rate : 0.6118
##   Detection Prevalence : 0.7176
##      Balanced Accuracy : 0.7676
##
##       'Positive' Class : neg
##
```

Accuracy of this model is better then that of the previous regression(0.8235).

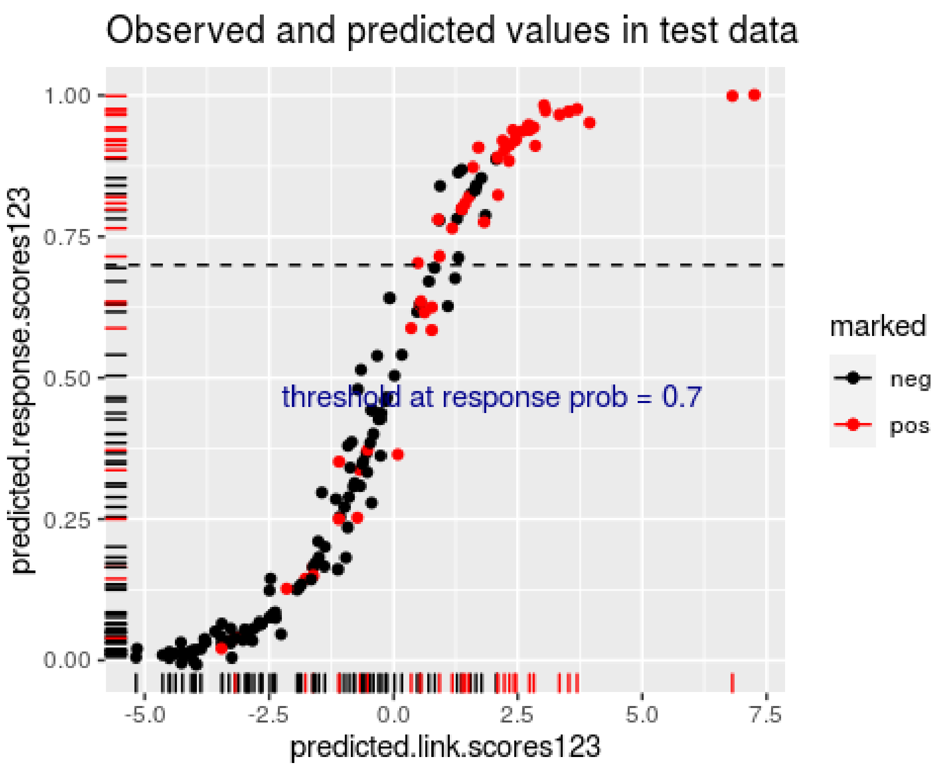
Let's add cosine distance3 values as the third predictor.

```
fit123 <- glm(manual_polarity~cosine_distance + cosine_distance2 +
cosine_distance3, data = train, family = "binomial")
fit123_full <- glm(manual_polarity~cosine_distance + cosine_distance2 +
cosine_distance3, data = senti_df, family = "binomial")
summary(fit123_full)

##
## Call:
## glm(formula = manual_polarity ~ cosine_distance + cosine_distance2 +
##     cosine_distance3, family = "binomial", data = senti_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9113  -0.5979  -0.2504   0.4997   2.4939
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.3746     0.3410  -1.099  0.27195
## cosine_distance    0.7847     3.3278   0.236  0.81359
## cosine_distance2   9.6220     3.0710   3.133  0.00173 **
## cosine_distance3   2.9517     2.8229   1.046  0.29573
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.48  on 280  degrees of freedom
## Residual deviance: 220.83  on 277  degrees of freedom
```

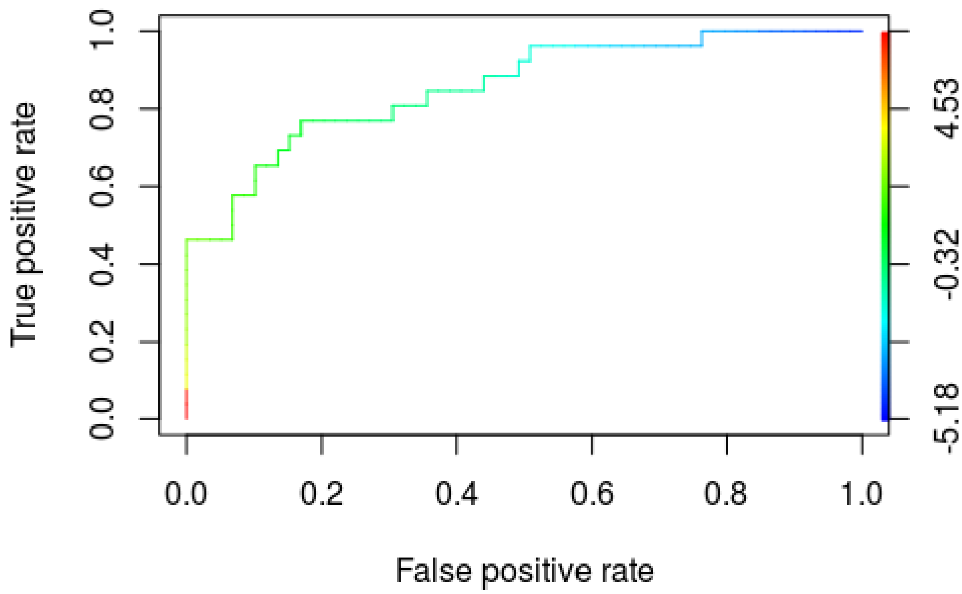
```
## AIC: 228.83
##
## Number of Fisher Scoring iterations: 5
```

This model shows lower AIC value(228.83), then model with two predictors(227.93), and higher, then regression with cosine_distance1 as predictor(237.84). Cosine_distance2 values again has higher influence on predictions(9.6220), than others.

```
predicted.link.scores123 <- predict(fit123, newdata=test, type="link")
predicted.response.scores123 <- predict(fit123, newdata=test,
type="response")
predicted.scores123 <- data.frame(link=predicted.link.scores123,
                         response=predicted.response.scores123,
marked=test$manual_polarity,
                         #construction_obs=load.test$CONSTRUCTION,
                         stringsAsFactors=FALSE)

predicted.scores123 %>%
  ggplot(aes(x=predicted.link.scores123 ,
y=predicted.response.scores123 , col=marked )) +
  scale_color_manual(values=c("black", "red")) +
  geom_point() +
  geom_rug() +
  geom_jitter(width=.7, height=.02) +
  geom_hline(yintercept=0.7, linetype="dashed") +
  annotate(geom="text", x=2, y=0.47, label="threshold at response prob
= 0.7", color="darkblue") +
  ggtitle("Observed and predicted values in test data")
```




```
pred123 <- predict(fit123, newdata=test)
predicted123 <- prediction(pred123, test$manual_polarity)
predicted123 <- performance(predicted123, "tpr", "fpr")
plot(predicted123, colorize=T)
```



Let’s choose prediction coefficient 0.7.

```
v <- rep(NA, nrow(predicted.scores123))
v <- ifelse(predicted.scores123$response >= .7, "pos", "neg")
predicted.scores123$tonality_pred <- as.factor(v)
#predicted.scores$tonality_pred
#predicted.scores$marked
cnf123 <- confusionMatrix(data = predicted.scores123$tonality_pred,
reference = predicted.scores123$marked)
cnf123

## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  53   9
##        pos   6  17
##
##                Accuracy : 0.8235
##                  95% CI : (0.7257, 0.8977)
##     No Information Rate : 0.6941
##     P-Value [Acc > NIR] : 0.005003
##
##                   Kappa : 0.5706
##
```

```
##  Mcnemar's Test P-Value : 0.605577
##
##             Sensitivity : 0.8983
##             Specificity : 0.6538
##          Pos Pred Value : 0.8548
##          Neg Pred Value : 0.7391
##              Prevalence : 0.6941
##          Detection Rate : 0.6235
##    Detection Prevalence : 0.7294
##       Balanced Accuracy : 0.7761
##
##        'Positive' Class : neg
##
```

Accuracy of this model is lower then previous regression's one(0.8118), and the same, as the accuracy of the model with cosine_distances1 as predictor(0.8235).

Let's measure in the same manner the accuracy and AIC metrices for regressions with cosine_distances 1 and 2 as single predictors

```
fit3<- glm(manual_polarity~ cosine_distance3, data = train, family =
"binomial")
fit3_full<- glm(manual_polarity~ cosine_distance3, data = senti_df,
family = "binomial")
summary(fit3_full)
```

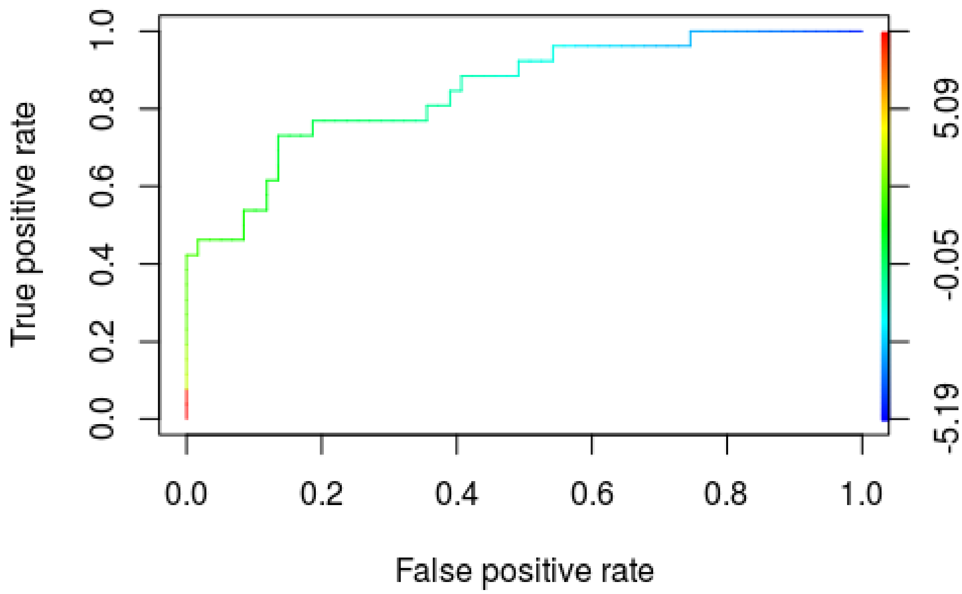
```
##
## Call:
## glm(formula = manual_polarity ~ cosine_distance3, family =
"binomial",
##     data = senti_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9292  -0.6240  -0.3363   0.5324   2.3438
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.3035     0.1973   1.539    0.124
## cosine_distance3   8.5275     1.0541   8.090 5.99e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.48  on 280  degrees of freedom
## Residual deviance: 234.65  on 279  degrees of freedom
## AIC: 238.65
```

```
##
## Number of Fisher Scoring iterations: 5
```

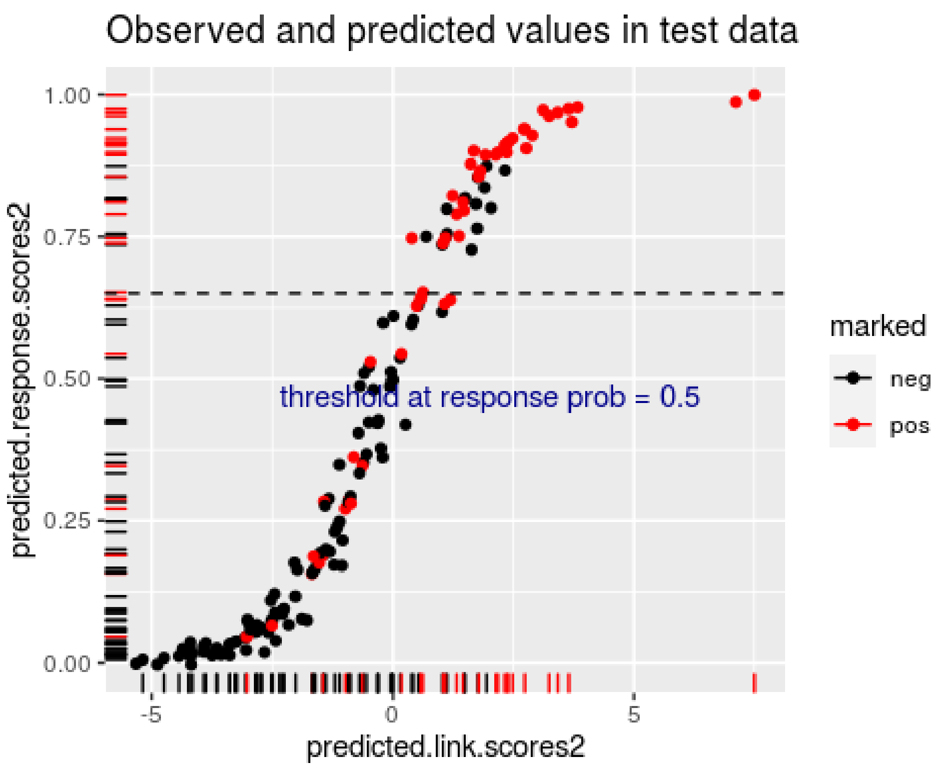
```
predicted.link.scores3<- predict(fit3, newdata=test, type="link")
predicted.response.scores3 <- predict(fit3, newdata=test,
type="response")
predicted.scores3<- data.frame(link=predicted.link.scores3,
                                response=predicted.response.scores3,
marked=test$manual_polarity,
                                #construction_obs=load.test$CONSTRUCTION,
                                stringsAsFactors=FALSE)
v <- rep(NA, nrow(predicted.scores3))
v <- ifelse(predicted.scores3$response >= .65, "pos", "neg")
predicted.scores3$tonality_pred <- as.factor(v)
#predicted.scores$tonality_pred
#predicted.scores$marked
cnf3 <- confusionMatrix(data = predicted.scores3$tonality_pred,
reference = predicted.scores$marked)
cnf3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  54  10
##        pos   5  16
##
##                Accuracy : 0.8235
##                  95% CI : (0.7257, 0.8977)
##     No Information Rate : 0.6941
##     P-Value [Acc > NIR] : 0.005003
##
##                   Kappa : 0.5608
##
##  Mcnemar's Test P-Value : 0.301700
##
##             Sensitivity : 0.9153
##             Specificity : 0.6154
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.7619
##              Prevalence : 0.6941
##          Detection Rate : 0.6353
##    Detection Prevalence : 0.7529
##       Balanced Accuracy : 0.7653
##
##        'Positive' Class : neg
##
```

```
fit2<- glm(manual_polarity~ cosine_distance2, data = train, family =
"binomial")
fit2_full<- glm(manual_polarity~ cosine_distance2, data = senti_df,
family = "binomial")
summary(fit2_full)

##
## Call:
## glm(formula = manual_polarity ~ cosine_distance2, family =
"binomial",
##     data = senti_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8148  -0.6211  -0.2635   0.5454   2.4451
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.7443     0.1692  -4.398 1.09e-05 ***
## cosine_distance2  14.5299     1.8530   7.841 4.46e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.48  on 280  degrees of freedom
## Residual deviance: 225.19  on 279  degrees of freedom
## AIC: 229.19
##
## Number of Fisher Scoring iterations: 5

predicted.link.scores2<- predict(fit2, newdata=test, type="link")
predicted.response.scores2 <- predict(fit2, newdata=test,
type="response")
predicted.scores2<- data.frame(link=predicted.link.scores2,
                         response=predicted.response.scores2,
marked=test$manual_polarity,
                         #construction_obs=load.test$CONSTRUCTION,
                         stringsAsFactors=FALSE)

pred2<- predict(fit2, newdata=test)
predicted2 <- prediction(pred2, test$manual_polarity)
predicted2 <- performance(predicted2, "tpr", "fpr")
plot(predicted2, colorize=T)
```




```
predicted.scores2 %>%
  ggplot(aes(x=predicted.link.scores2, y=predicted.response.scores2 ,
col=marked )) +
  scale_color_manual(values=c("black", "red")) +
  geom_point() +
  geom_rug() +
  geom_jitter(width=.7, height=.02) +
  geom_hline(yintercept=0.65, linetype="dashed") +
  annotate(geom="text", x=2, y=0.47, label="threshold at response prob
= 0.5", color="darkblue") +
  ggtitle("Observed and predicted values in test data")
```



Confusion matrix.

```
v <- rep(NA, nrow(predicted.scores2))
v <- ifelse(predicted.scores2$response >= .7, "pos", "neg")
predicted.scores2$tonality_pred <- as.factor(v)
#predicted.scores$tonality_pred
#predicted.scores$marked
cnf2 <- confusionMatrix(data = predicted.scores2$tonality_pred,
reference = predicted.scores123$marked)
cnf2

## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg  51  10
##        pos   8  16
##
##                Accuracy : 0.7882
##                  95% CI : (0.6861, 0.8694)
##     No Information Rate : 0.6941
##     P-Value [Acc > NIR] : 0.03552
##
##                   Kappa : 0.4903
##
```

```
##  Mcnemar's Test P-Value : 0.81366
##
##             Sensitivity : 0.8644
##             Specificity : 0.6154
##          Pos Pred Value : 0.8361
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6941
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.7176
##       Balanced Accuracy : 0.7399
##
##        'Positive' Class : neg
##
```

Accuracy of this model looks higher then the others.

Considering strong linear connection between all cosine distance values it might be useful to take a brief look at the Principal Component Analysis method. We can transform the data with this method and fit regression with the transformed cosine distance 1, 2 and 3 variables:

```r
cosine.pca <- prcomp(senti_df[4:6], scale = TRUE)
fit_pca123<- glm(senti_df$manual_polarity~ cosine.pca$x[,1] +
cosine.pca$x[,2] + cosine.pca$x[,3], data = train, family = "binomial")
summary(fit_pca123)

##
## Call:
## glm(formula = senti_df$manual_polarity ~ cosine.pca$x[, 1] +
##     cosine.pca$x[, 2] + cosine.pca$x[, 3], family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9113  -0.5979  -0.2504   0.4997   2.4939
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.4147     0.2072  -6.829 8.53e-12 ***
## cosine.pca$x[, 1]  -1.2188     0.1546  -7.884 3.16e-15 ***
## cosine.pca$x[, 2]  -0.7932     0.4846  -1.637    0.102
## cosine.pca$x[, 3]   0.3986     0.8321   0.479    0.632
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.48  on 280  degrees of freedom
```

```
## Residual deviance: 220.83  on 277  degrees of freedom
## AIC: 228.83
##
## Number of Fisher Scoring iterations: 5
```

Now the difference in the influence of cosine distances on the predictions is not so strong, but AIC coefficient is the same as the coefficient of regression with three predictors fitted without PCA method.

The table below shows the summarized results of the accuracy and AIC values of the models.

```
models <- c("Model1", "Model2", "Model3", "Model12", "Model123",
"PCA_Model123")

aics <- c(extractAIC(fit1_full)[2], extractAIC(fit2_full)[2],
extractAIC(fit3_full)[2], extractAIC(fit12_full)[2],
extractAIC(fit123_full)[2], extractAIC(fit_pca123)[2])

accs <- c(cnf$overall[['Accuracy']], cnf2$overall[['Accuracy']],
cnf3$overall[['Accuracy']], cnf12$overall[['Accuracy']],
cnf123$overall[['Accuracy']], "-")

data.frame(models, accs, aics)

##         models              accs     aics
## 1       Model1 0.823529411764706 237.8366
## 2       Model2 0.788235294117647 229.1912
## 3       Model3 0.823529411764706 238.6505
## 4      Model12 0.811764705882353 227.9312
## 5     Model123 0.823529411764706 228.8288
## 6 PCA_Model123                 - 228.8288

4.1 Summary
```

According to the results, model12 has the best aic, while model2 has higher accuracy. It is also interesting, that cosine_distances2 values have stronger influence on the predictions of the regressions in combination with other cosine values. The difference between the models is not so strong, but according to the best AIC, we can use Model2 in larger datasets. It has also been shown, that there is no connection between manual polarity and tonality annotations made by different people.

References.

Kutuzov A., Kuzmenko E. (2017) WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models. In: Ignatov D. et al. (eds) Analysis of Images, Social Networks and Texts. AIST 2016. Communications in Computer and Information Science, vol 661. Springer, Cham

Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright c 2019. Draft of October 2, 2019.

Distant Supervision for Sentiment Attitude Extraction. Nicolay Rusnachenko, Natalia Loukachevitch, Elena Tutubalina, 2016.

Creating a General Russian Sentiment Lexicon Natalia Loukachevitch, Anatoly Levchik Lomonosov Moscow State University, 2016.

Evaluating Word Embedding Models: Methods and Experimental Results Bin Wang∗ , Student Member, IEEE, Angela Wang∗ , Fenxiao Chen, Student Member, IEEE, Yuncheng Wang and C.-C. Jay Kuo, Fellow, IEEE