|  |  |
| --- | --- |
|  | **Technical description of**  **ASAP3 Engine for Smarter Balanced short answer scoring**  29 January, 2014 |

**Author:** Xavier Conort, Steve Harbick

**Email:** [xavier.conort@gear-analytics.com](mailto:xavier.conort@gear-analytics.com), [steve.harbick@signal2meaning.com](mailto:steve.harbick@signal2meaning.com)

**Competition:** ASAP3 – Short Constructed Response Scoring

1. **Response cleaning and pre-processing rules**

**File Processing**

A Python script was created to decompress the zip files into XML (using zipfile), then parse the XML trees (using lxml) into a Python pandas DataFrame (using pandas). Pandas DataFrame was chosen to enable further manipulation of the data using Python libraries; but for short constructed response, no further processing was done. Several working files were then saved:

* tabbed-delimited flat file for further processing in R
* pickle file for quick loading of data back into Python, if required
* original compressed zip files
* decompressed XML files

Example for item 3\_51802\_TS\_51802\_1

* 3\_51802\_TS\_51802\_1.tsv
* 3\_51802\_TS\_51802\_1.pickle
* 3\_51802\_TS\_51802.zip
* 3\_51802\_TS\_51802\_1.xml

**Response cleaning**

Responses containing a string with more than 45 characters have been replaced by “cheating”.

Example:

“yes!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"

**Text Statistics**

Various text statistics were computed and added as features in some of our regression models:

* size of longest string in the response
* number of words
* number of unique words
* ratio number of unique words / number of words
* number of characters
* statistics on the words’ length (quantiles 90%, 95% and 99%)
* number of sentences
* statistics on number of words per sentence (mean, min, max, std)
* number of digits
* number of punctuations, quotes, slash, exclamation, colon, question mark, bracket, comma, and semicolon divided by the number of words
* percentage of transition words based on a list of transition words; n-grams (1 through 4-grams) present in the list of transition words were flagged as transition words
* percentage of precise verbs, based on a list of stemmed precise verbs; stemmed n-grams (1 through 2-grams) present in the list of stemmed precise verbs were flagged as transition words
* percentage of misspelling

Example:

Raw text: [33] "she is ronge. two/threeds is grater then two/four."

Proxies generated:

|  |  |
| --- | --- |
| string\_max | 11.00 |
| NbWords | 10.00 |
| UniqueWords | 8.00 |
| RUniqueWords | 1.25 |
| nchar | 50.00 |
| WordsLengthQ90 | 6.10 |
| WordsLengthQ95 | 6.55 |
| WordsLengthQ99 | 6.91 |
| NbSentences | 2.00 |
| WordsMeanInSent | 5.00 |
| WordsSdInSent | 1.41 |
| WordsMaxInSent | 5.00 |
| WordsMinInSent | 3.00 |
| Digit | 0.00 |
| Punct | 0.30 |
| QuotM | 0.00 |
| Slash | 0.20 |
| Emark | 0.00 |
| Colon | 0.00 |
| Question | 0.00 |
| Braket | 0.00 |
| Comma | 0.00 |
| SemiColon | 0.00 |
| PTransW | 0.10 |
| PPrV | 0.00 |
| PErr | 0.20 |

One working RData file for each item is generated to save the training proxies features and stored in the folder “Working\_files”.

Example for item 3\_51802\_TS\_51802\_1

* V1\_3\_51802\_TS\_51802\_1\_PROXIES.RData
* For the test set, the prefix is replaced by V1\_H

**Spelling errors count**

This was a very simple method. A spelling error was counted every time a word didn’t appear in a list of “correct words” generated from various sources:

* norvig.txt from <http://norvig.com/big.txt> (concatenation by Peter Norvig of several public domain books from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus) to estimate spelling errors.
* gutenberg.txt Ebooks from <http://gutenberg.org> (Creation Myths of Primitive America, by Jeremiah Curtin; The Legend of Sleepy Hollow, by Washington Irving; Tender Buttons, by Gertrude Stein; Three Soldiers, by John Dos Passos) to estimate spelling errors.
* essay\_inst.txt from combination of the 10 sets instructions to estimate spelling errors.
* academic words.csv from <http://www.uefap.com/vocab/select/awl.htm> to estimate spelling errors.
* my\_list.txt from my own list generated for the long essay competition which has not been updated for the short answer competition to estimate spelling errors.

When a word was flagged as a spelling error and appears multiple times in the document, it was counted only once.

Example:

*"The information I would need in order to sucessfully replicate the experiment is the correct measurement they used for the experiment, also the materials that was used to creat the experiament. After the 24 hours, of removing the samples from the container and rinsen each sample with distilled water. Making it dry for 30 mintued might not be long enough to really determine the mass of each sample, especially when you have more than one different samples to do with different texture."*

Spelling errors detected = "creat", "experiament", "mintued", "rinsen", "sucessfully"

**Numbers**

We converted numbers into words.

Example: “1” was replaced by “one”. “25” was replaced by “twofive”.

**N-grams**

Word n-grams and Character n-grams were produced using the function textcnt, from the R package tau, that automatically removes punctuation and converts words into lower case.

Word n-grams:

From raw text, produced 1-gram, 2-grams, 3-grams; and then stemmed them using the function wordStem from the R package RTextTools.

Example:

*“she is ronge. two/threeds is grater then two/four”*

1-gram:

four, grater, is, ronge, she, then, threeds, two

2-grams:

grater then, is grater, is ronge, ronge two, she is, then two, threeds is, two four, two threeds

3-grams:

grater then two, is grater then, is ronge two, ronge two threeds, she is ronge, then two four, threeds is grater, two threeds is

Character n-grams:

From raw text (example in appendix): Produced 4-grams and 6-grams. The presence of any punctuation was used as a splitting rule.

Example:

*“she is ronge. two/threeds is grater then two/four”*

4-grams:

, g, gr, gra, i, is, is , r, ro, ron, t, th, the, tw, two, \_, \_ , \_ t, \_ tw, \_f, \_fo, \_fou, \_s, \_sh, \_she, \_t, \_th, \_thr, a, at, ate, ater, d, ds, ds , ds i, e, e , e i, e is, e\_, ed, eds, eds , ee, eed, eeds, en, en , en t, er, er , er t, f, fo, fou, four, g, ge, ge\_, gr, gra, grat, h, he, he , he i, hen, hen , hr, hre, hree, i, is, is , is g, is r, n, n , n t, n tw, ng, nge, nge\_, o, o\_, on, ong, onge, ou, our, our\_, r, r , r t, r th, r\_, ra, rat, rate, re, ree, reed, ro, ron, rong, s, s , s g, s gr, s i, s is, s r, s ro, sh, she, she , t, te, ter, ter , th, the, then, thr, thre, tw, two, two\_, u, ur, ur\_, w, wo, wo\_

6-grams:

, g, gr, gra, grat, grate, i, is, is , is g, is gr, is r, is ro, r, ro, ron, rong, ronge, t, th, the, then, then , tw, two, two\_, \_, \_ , \_ t, \_ tw, \_ two, \_ two\_, \_f, \_fo, \_fou, \_four, \_four\_, \_s, \_sh, \_she, \_she , \_she i, \_t, \_th, \_thr, \_thre, \_three, a, at, ate, ater, ater , ater t, d, ds, ds , ds i, ds is, ds is , e, e , e i, e is, e is , e is r, e\_, ed, eds, eds , eds i, eds is, ee, eed, eeds, eeds , eeds i, en, en , en t, en tw, en two, er, er , er t, er th, er the, f, fo, fou, four, four\_, g, ge, ge\_, gr, gra, grat, grate, grater, h, he, he , he i, he is, he is , hen, hen , hen t, hen tw, hr, hre, hree, hreed, hreeds, i, is, is , is g, is gr, is gra, is r, is ro, is ron, n, n , n t, n tw, n two, n two\_, ng, nge, nge\_, o, o\_, on, ong, onge, onge\_, ou, our, our\_, r, r , r t, r th, r the, r then, r\_, ra, rat, rate, rater, rater , re, ree, reed, reeds, reeds , ro, ron, rong, ronge, ronge\_, s, s , s g, s gr, s gra, s grat, s i, s is, s is , s is g, s r, s ro, s ron, s rong, sh, she, she , she i, she is, t, te, ter, ter , ter t, ter th, th, the, then, then , then t, thr, thre, three, threed, tw, two, two\_, u, ur, ur\_, w, wo, wo\_

**Document Term Matrix**

For each item, we produced three incidence document term matrices that are binary matrices indicating presence/absence of a term in an answer.

1. One is based on stemmed words n-grams (1-gram through 3-grams).
2. One is based on character 4-grams.
3. One is based on character 6-grams.

Three working RData files are generated for the training set and each item.

Example for item 3\_51802\_TS\_51802\_1

* V1\_3\_51802\_TS\_51802\_1\_DTM\_Words\_3grams.RData
* V1\_3\_51802\_TS\_51802\_1\_DTM\_Chars\_4grams.RData
* V1\_3\_51802\_TS\_51802\_1\_DTM\_Chars\_6grams.RData

For the test set, the prefix is replaced by V1\_H

1. **Rules used to identify papers not eligible for AI scoring (e.g., too short, too long, different language)**

Records with non-numeric scores have been removed from the training set.

1. **Methods for dictionary building (as appropriate):**

To estimate the usage of precise verbs, transition words, and spelling errors, dictionaries were generated from various sources:

* precise\_verbs.csv from <http://www.owlnet.rice.edu/~cainproj/writingtips/preciseverbs.html> to estimate usage of precise verbs.
* transition\_words.csv from <http://www.smart-words.org> to estimate usage of transition words.
* norvig.txt from <http://norvig.com/big.txt> (concatenation by Peter Norvig of several public domain books from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus) to estimate spelling errors.
* gutenberg.txt Ebooks from <http://gutenberg.org> (Creation Myths of Primitive America, by Jeremiah Curtin; The Legend of Sleepy Hollow, by Washington Irving; Tender Buttons, by Gertrude Stein; Three Soldiers, by John Dos Passos) to estimate spelling errors.
* essay\_inst.txt from combination of the 10 sets instructions to estimate spelling errors.
* academic words.csv from <http://www.uefap.com/vocab/select/awl.htm> to estimate spelling errors.
* my\_list.txt from my own list generated for the long essay competition which has not been updated for the short answer competition to estimate spelling errors.

1. **Methods for feature selection:**

We operated three types of feature selection

* We discarded rare n-grams (appearing less than 15 times per item)
* We trained:
  + Random Forest on character n-grams document term matrices, obtained the permutation importance of each term and selected terms with the highest importance. The threshold was set in order to have the total importance unchanged. As a result, terms with small positive importance and negative importance were discarded and the sum of the importance of the discarded terms is equal to 0.
  + Regularized Regression on word n-grams document term matrix and selected terms that had non-null coefficients.
* We applied Principal Component Analysis (PCA) on the reduced matrices.

5 RData files to save the training results from Random Forest, Regularized regression and PCA are stored in the folder “Working\_files”.

Example for item 3\_51802\_TS\_51802\_1

* V1\_3\_51802\_TS\_51802\_1\_GLMNET\_4fs\_DTM\_Words\_3grams.RData
* V1\_3\_51802\_TS\_51802\_1\_RF\_4fs\_DTM\_Chars\_4grams.RData
* V1\_3\_51802\_TS\_51802\_1\_RF\_4fs\_DTM\_Chars\_6grams.RData
* V1\_3\_51802\_TS\_51802\_1\_PCA\_Model\_Small\_DTM\_Chars\_4grams.RData
* V1\_3\_51802\_TS\_51802\_1\_PCA\_Model\_Small\_DTM\_Chars\_6grams.RData

5 RData files containing reduced document term matrices are generated for the training set and each item.

Example for item 3\_51802\_TS\_51802\_1

* V1\_3\_51802\_TS\_51802\_1\_Small2\_DTM\_Words\_3grams.RData
* V1\_3\_51802\_TS\_51802\_1\_Small\_DTM\_Chars\_4grams.RData
* V1\_3\_51802\_TS\_51802\_1\_Small\_DTM\_Chars\_6grams.RData
* V1\_3\_51802\_TS\_51802\_1\_PCA\_Small\_DTM\_Chars\_4grams.RData
* V1\_3\_51802\_TS\_51802\_1\_PCA\_Small\_DTM\_Chars\_6grams.RData

The files for the test set use the prefix ”V1\_H”

1. **Methods for predictions:**

Our modeling approach consists of training a set of base learners on the different document term matrices, and then combining them with stacking.

**Individual Models**

For each item, we trained 22 individual models using the following Machine Learning algorithms, all available in R:

1. 4 Random Forest models (RF) (randomForest package) trained on
   1. Character 4 grams matrix
   2. Character 6 grams matrix
   3. Reduced Word (1through3) grams matrix
   4. Reduced Word (1through3) grams matrix + PROXIES
2. 2 Gradient Boosting Machine models (GBM) with Gaussian error (gbm package) trained on
   1. Reduced Word (1through3) grams matrix
   2. Reduced Word (1through3) grams matrix + PROXIES
3. 6 Regularized Generalized Linear Models with Gaussian Error (glmnet packages) trained on
   1. Word (1through3) grams matrix
   2. Reduced Word (1through3) grams matrix
   3. Reduced Character 4 grams matrix
   4. Reduced Character 6 grams matrix
   5. Reduced Character 4 grams matrix transformed with PCA
   6. Reduced Character 6 grams matrix transformed with PCA
4. 5 Support Vector Machine (SVM) with linear kernel (e1071 package) trained on
   1. Reduced Word (1through3) grams matrix
   2. Reduced Character 4 grams matrix
   3. Reduced Character 6 grams matrix
   4. Reduced Character 4 grams matrix transformed with PCA
   5. Reduced Character 6 grams matrix transformed with PCA
5. 5 Support Vector Machine (SVM) with radial kernel (e1071 package) trained on
   1. Reduced Word (1through3) grams matrix
   2. Reduced Character 4 grams matrix
   3. Reduced Character 6 grams matrix
   4. Reduced Character 4 grams matrix transformed with PCA
   5. Reduced Character 6 grams matrix transformed with PCA

Tuning parameters for SVM are fine-tuned with a home-made pattern search (code available in CV\_SVM\_lin.R and CV\_SVM\_rad.R).

The number of trees for GBM is set with early stopping (code available CV\_GBM.R).

For all our models, we followed a 10 folds cross-validation approach.

Predictions for the test set are an average of the predictions obtained from the 10 models trained on the 10 partitions of the cross-validation procedure.

**Ensemble by Stacking**

To blend different fits together, the holdout predictions of the 22 models were given as input to a second-level learning algorithm, an approach known as “stacking”. The algorithms used at the second-level is a non-negative least squares regression (nnls package in R).

**Conversion of Predictions to optimize Kappa**

We adjusted our predictions to optimize our Kappa score. 1d polynomial was used as the framework for the adjustment. To achieve this, we maximized the following functions using an optimization algorithm based on Nelder–Mead (default of the R function optim):

* SQWKappa(y,round(xx[1]+xx[2]\*x,0))

Where SQWKappa is the function to compute the quadratic weighted Kappa, y the score, x the non-adjusted predictions and xx the coefficients to estimate.