The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–13

Estimating Text Regressions using `txtreg_train`

Carlo Schwarz
Bocconi University
Milano, Italy
carlo.schwarz@unibocconi.it

Abstract. This paper introduces new Stata commands to estimate text regressions for continuous, binary, and categorical variables based on text strings. The `txtreg_train` command automatically handles text cleaning, tokenization, model training, and cross-validation for LASSO, Ridge, ElasticNet, and regularized logistic regressions. The `txtreg_predict` command allows to obtain the predictions from the trained text regression model. Further, the analysis of the coefficients of the text regression model is facilitated by the `txtreg_analyze` command. Together these commands provide a convenient toolbox for researchers to train text regressions. They also allow sharing of pre-trained text regression models with other researchers.

Keywords: st0001, text regressions, machine learning, text analysis

1 Introduction

In recent years natural language processing (NLP) has risen to increased prominence in the social sciences. This rise was not only driven by increases in computing power and data availability but also by the enormous amount of previously unaccessible information that is contained in text form. A widely used approach in NLP are so-called text regressions (see Gentzkow et al. 2019, for a discussion of papers). Text regressions use the words in texts as independent variables ($\mathbf{X}$) to predict an outcome (y). The great flexibility of this approach comes from the fact that it is applicable to any form of text data and allows for the automatic prediction of outcome variables. Text regressions are also often used to impute variables in new data sets. It is, for example, possible to hand-code a subset of data and then extend the coding to the entire data set. Alternatively, text regressions can learn relationships between an outcome and text in one data set and create a new outcome variable for other data for which this information is lacking.

This paper introduces three Stata commands for text regressions. First, `txtreg_train` allows to train text regression models and then store them for later use or for sharing with other researchers. Secondly, the `txtreg_predict` command handles the prediction of outcomes based on pre-trained models. Lastly, `txtreg_analyze` facilitates the analysis of the coefficients of the text regression model. This enables the investigation of relationships between words and the outcome variable the text regression has derived.

In this way, the commands extend the NLP capabilities of Stata. Among others, Stata already can calculate the Levenshtein distance (Barker 2012), perform Latent Dirichlet Allocation (Schwarz 2018), and Latent Semantic Analysis (Schwarz 2019).

The closest related commands are the `txttool` command by Williams and Williams (2014) and the `ngram` command by Schonlau et al. (2017) both of which can tokenize text for use in statistical analysis. The presented commands provide at least three advantages. First, they utilize Stata's Python integration, which was introduced in Stata 16 and enables the use of highly optimized Python packages.[1] Second, the commands provide an integrated pipeline that should be suitable for the most frequent uses of text regressions. Last and most importantly, the commands provide an easy way to store trained text regression models. As the extrapolation of outcome variables beyond the training data is one of the most frequent uses of text regressions, this opens up new applications. The stored text regression models can also be shared with other researchers without sharing the underlying raw data.

The rest of the paper proceeds as follows. Section 2 provides a short introduction to text regressions. Section 3 describes the Stata commands followed by an example that uses text regressions for the prediction of citations (see Section 4). Section 5 concludes.

2 Predicting Numerical Variables using Text

Consider a setting in which a researcher wants to predict a dependent variable (y) based on information contained in a string variable. Text regressions approach this problem by using the words in the text strings as the independent variables ($\mathbf{X}$). More specifically, text regressions minimize the mean squared deviation from a dependent variable y based on the observations $i \in N$:

$$\min_{\beta_j \in W} C = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{W} \beta_j \cdot w_{ij} \right)^2 \tag{1}$$

where w_{ij} are the individual text features, and β_j are the regression coefficients of the text regression. The text features are constructed from the text string by creating a bag-of-words representation, which represents the text as a document-n-gram-matrix $\mathbf{W}$. The term n-grams refers to continuous sequences of n-words from the original text string, where n indicates the length of the sequence. Hence, 1-grams (unigrams) are simply the set of words that appear in the text. Similarly, 2-grams (bigrams) and 3-grams (trigrams) are the set of 2 and 3 words phrases that appear in the text. The dimensions of the matrix $\mathbf{W}$ are $N \times V$, where N is the number of observations and V is the size of the vocabulary (i.e., the number of unique n-grams). The entries $f_{i,v}$ in

1. As the commands make use of Stata's Python integration, they require at least Stata 16 to run. The commands further require a working Python installation. A Python installation that includes the required packages is, for example, Anaconda. In particular, the commands rely on the Scikit (Pedregosa et al. 2011), Pandas (The Pandas Development Team 2020), and NLTK (Bird et al. 2009) packages.

W represent the number of times a n-gram $v \in V$ appears in observation $i \in N$:

$$\underbrace{W}_{N \times V} = \begin{pmatrix} f_{1,1} & \cdots & f_{1,V} \\ \vdots & \ddots & \vdots \\ f_{N,1} & \cdots & f_{N,V} \end{pmatrix}$$

As the matrix **W** will form the main input for the text regression, it is, of course, important to decide which n-grams should be included in the matrix. The following three steps have become standard in text regressions. First, very frequent words (stopwords) that add little to the meaning of texts (e.g., "I", "he", "and") are removed upfront. Frequently, also thresholds are used for the minimum and the maximum number of occurrences of n-grams across different observations. Secondly, words are reduced to their morphological roots using a process called stemming. For example, stemming will change "walking" and "walked" to "walk", such that these words represent the same unigram. Lastly, the matrix **W** is often reweighted using term frequency-inverse document frequency (tf-idf), which replaces $f_{i,v}$ by $tfidf(f_{i,v}) = (1 + \log(f_{i,v})) \cdot \left(\log\left(\frac{1+N}{1+n_v}\right) + 1\right)$, where n_v is the number of observations in which the n-gram v appears. The tf-idf reweighting reduces the weights of n-grams that appear in many observations and therefore contribute less to the meaning of texts.

While we could estimate a text regression using Equation 1, it should be apparent that the dimensions of matrix **W** can become very large due to the growing size of the vocabulary, in particular when trigrams are used. It is also not rare that $V > N$ which makes it impossible to estimate an OLS regression. Even in cases where $V < N$, it might not be advisable to estimate an OLS regression as the regression might fit idiosyncratic features of the training data. To overcome these limitations, the machine learning literature makes use of regularization and k-fold cross-validation (see Hastie et al. 2001, for additional details). Regularization implies the inclusion of an additional term in the cost function C (see Equation 1) which incorporates a penalty for the size of the coefficients β_j. The most frequently used regularized regressions models and their respective penalties are LASSO ($\lambda \sum_{j=1}^{W} |\beta_j|$), Ridge ($\lambda \sum_{j=1}^{W} \beta_j^2$) and ElasticNet ($\lambda_1 \sum_{j=1}^{W} |\beta_j| + \lambda_2 \sum_{j=1}^{W} \beta_j^2$). The user-chosen parameters λ_1 specify the strength of regularization.[2] The optimal regularization strength, in turn, is chosen using k-fold cross-validation. This process involves randomly splitting the data into k slices. The model is then trained on $k-1$ slices, and the model's performance is evaluated on the remaining data slice, e.g., the model's R^2. The model with the best out-of-sample performance determines the optimal regularization strength. In other words, the regularization strength is chosen in such a way that the text regression model best explains the variation in unseen data.

2. In cases where the dependent variable y is binary or categorical, it is similarly possible to include regularization terms in logistic or multinomial logistic regression models.

3 Stata Implementation

This section describes how the `txtreg_*` commands automate the training of text regressions in Stata. If no pre-trained model is used, the user should start with the `txtreg_train` command.

Training of Text Regression using `txtreg_train`

The `txtreg_train` command handles the training of text regression models. The user needs to specify the dependent and independent variables. The dependent variable can be continuous, binary, or categorical, while the independent variable should always be a string variable. The rest of the training process is automatically handled by the command. In particular, the command performs the following steps:

1) The command cleans the text and creates the document-n-gram-matrix. If specified, the command will remove stopwords, stemm the text, and use tf-idf reweighting.
2) Afterwards, the data are split into a training sample (90%) and a test sample (10%).
3) The text regression model is trained on the training sample, and the optimal regularization strength is chosen using 10-fold cross-validation.
4) The command reports the optimal regularization strength and the out-of-sample performance of the text regression based on the test sample.

The provided options allow to control the text preparation process and the training of the model.

Syntax

`txtreg_train` *depvar indepvar* `using` *filename* [`,` `model(`*string*`)`
`regularization(`*string*`)` `regu_range(`*string*`)` `seed(`*integer*`)` `tfidf` `stem`
`stem_lang(`*string*`)` `stopwords(`*string*`)` `ngrams(`*integer*`)` `min_freq(`*integer*`)`
`max_freq(`*real*`)` `max_voc(`*integer*`)`]

Options

`using` *filename* specifies the path and the name for storing the trained text regression model. If `using` is not specified a file called "txtreg_model.pkl" will be generated in the current working directory.

`model(`*string*`)` specifies if a least squares or logistic regression will be used to train the model. Least squares regression is used by specifying "reg". To use logistic

regression, specify "logit". By default, a least squares regression will be used. The command will automatically train a multinomial logistic model if the dependent variable is categorical and "logit" is specified.

`regularization(`*string*`)` specifies which form of regularization should be used. The options are either "lasso", "ridge", or "elasticnet". By defaulft "ridge" regularization will be used.

`regu_range(`*string*`)` specifies which values of the regularization parameter will be tested in the 10-fold cross-validation. The string should give the start and the end value separated by a comma. The default is `regu_range("0.1,1")` such that values between 0.1 and 1 in steps of 0.1 are used. In least squares regressions, larger values imply stronger regularization. For logistic regression, smaller values imply stronger regularization.[3]

`seed(`*integer*`)` specifies the seed for the random number generator, which among others, determines the splits for the cross-validation.

Text Cleaning Options

`tfidf` specifies if term-frequency-inverse-document-frequency (tf-idf) should be used. If the option is used, the document-n-gram-matrix is reweighted by term frequency-inverse document frequency before training the text regression model.

`stem` specifies if the words should be stemmed before estimating the text regression model. This will reduce the words to their morphological roots (e.g., walked to walk). The stemming implementation relies on Pythons NLTK package (Bird et al. 2009).

`stem_lang(`*string*`)` this option is only needed if `stem` is used. `stem_lang(`*string*`)` specifies the language of the text strings, such that the appropriate stemmer can be used. For a list of supported languages see the NLTK Website. The default language is `stem_lang("english")`.

`stopwords(`*string*`)` specifies a list of words to exclude from the text regression. Usually, highly frequent words such as "I", and "you" are removed from the text since these words do not help with the classification of the documents. Predefined stopword lists for different languages are available online (see, for example, here).

`ngrams(`*integer*`)` specifies which order of n-grams should be included in the text regression. For example, specifying `ngrams(2)` implies the use of unigrams and bigrams. `ngrams(3)` additionally uses trigrams. By default only unigrams are used.

`min_freq(`*integer*`)` allows the removal of words that appear in few documents. Words that appear in fewer documents than `min_freq(`*integer*`)` will be excluded from the text regression. The default is `min_freq(0)`.

3. In the case of elasticnet regularization, the two λ parameters are set jointly through the use of a mixing parameter. The ratio of lasso to ridge regularization is set to 0.5 by default.

`max_freq(real)` allows the removal of words that appear frequently in documents. Words that appear in a share of more than `max_freq(real)` documents will be excluded. The default is `max_freq(1)`.

`max_voc(integer)` allows to specify the maximum number of n-grams to be included in the text regression. By default, the vocabulary will not be restricted.

Output

`txtreg_train` generates a new file containing the trained text regression model. The filename and path are specified in `using`. If `using` is not specified, a file called "txtreg_model.pkl" will be generated in the current working directory.

Making use of Trained Text Regression Models with `txtreg_predict`

After the users either trained a text regression model using `txtreg_train` or obtained a pre-trained text regression model, the users can use the `txtreg_predict` command to perform the prediction of the outcome variable. For the prediction of the outcome, `txtreg_predict` makes use of text strings as the independent variable. The advantage of this separation between the training and prediction process is that 1) it is possible to apply the predicted citation model to a different data set than the training data, and 2) it is possible to share the trained model with other researchers. The syntax of `txtreg_predict` is:

Syntax

`txtreg_predict` *`indepvar`* `using` *`filename`* [`, name_new_var(`*string*`) stem stem_lang(`*string*`)`]

Options

`using` *`filename`* specifies the location and the name of the pre-trained text regression model.

`name_new_var(`*string*`)` specifies the name of the variable created by `txtreg_predict`. The user should ensure that `name_new_var` is not yet used in the data set. If nothing is specified, the name of the variable is "predict_".

`stem` and `stem_lang(`*string*`)` should be specified if the used text regression model stemmed the words before training. If it is unknown if the model is based on stemmed words, the `txtreg_analyze` command allows to investigate the n-grams on which the model is based. Stemming can easily be recognized from the n-grams.

Output

`txtreg_predict` creates a new variable with the name specified by `name_new_var(`*string*`)` containing the predictions from the text regression model.

Analyzing Text Regression with `txtreg_analyze`

When using text regressions, it is important to ensure that the derived relationships between n-grams and the outcome are sensible. This can be achieved by analyzing the coefficients of the text regression model. This is facilitated by the `txtreg_analyze` command. The syntax is:

Syntax

`txtreg_analyze using` *`filename`*

Output

`using` *`filename`* specifies the location and the name of the pre-trained text regression model. `txtreg_analyze` replaces the data set that is currently in memory. The new data set contains the n-grams and the estimated coefficients from the text regression.

4 Example: Predicting Citations based on Paper Titles

This section provides an example of the use of text regressions for the prediction of citations of scientific papers. This example is motivated by Iaria et al. (2021), who use predicted citations to control for observable differences between male and female-authored scientific papers to measure gender gaps in citations. First, the authors train a text regression model based on the titles and the citations of papers written by men. They then create a measure of predicted citations for all papers, i.e., how many citations we would expect a paper to receive based on its title if it was written by a man. Finally, this predicted citation measure is used to control for differences in citations between male and female-authored papers that occur due to the fact that women potentially work on different topics.

Model Training The example data set consists of 100,000 observations. Each observation represents one scientific paper with its title and the number of citations. First, the data are loaded, and the citation variable is transformed in two different ways (log and binary). Afterwards the `txtreg_train` command is called. The example shows the use of `txtreg_train` to train least squares and logistic regressions with different regularization terms.

```
. ********************************************************************
. **#* 1) Load Data
. ********************************************************************
.
. use "$path/Data/WoS_all_cite_prediction_example.dta"
.
. * generate log citations for regression
. gen WOS_TOTAL_ln = ln(1+WOS_TOTAL)
.
. * generate dummy for logit
. gen WOS_TOTAL_d = WOS_TOTAL>0
.
.
. * define list of stopwords
. global stopwords "a able about across after all almost also am among an and any ar
> e as at be because been but by can cannot could dear did do does either else ever
> every for from get got had has have he her hers him his how however i if in into i
> s it its just least let like likely may me might most must my neither no nor not o
> f off often on only or other our own rather said say says she should since so some
>  than that the their them then there these they this tis to too twas us wants was
> we were what when where which while who whom why will with would yet you your"
.
.
. ********************************************************************
. **#* 2) Train text regression using txtreg_train
. ********************************************************************
.
. **#* 2a)
. txtreg_train WOS_TOTAL_ln title using "$path/Models/predicted_citation_lasso.pkl"
>  , model("reg") regularization("lasso") regu_range(0.0001,0.001) ngrams(2) seed(15
> 02) tfidf stem stem_lang("english") stopwords("$stopwords") min_freq(3) max_freq(0
> .3) max_voc(5000)
Step 1/4 :Loading Data from Stata
Stemming text
Step 2/4 :Tokenizing Data
Tfidf is used
Step 3/4 :Training Model (This may take some time)
Score Fold 0 (Regularization=0.000050): 0.120290
Score Fold 1 (Regularization=0.000056): 0.132191
Score Fold 2 (Regularization=0.000061): 0.109037
Score Fold 3 (Regularization=0.000067): 0.117784
Score Fold 4 (Regularization=0.000072): 0.123905
Score Fold 5 (Regularization=0.000078): 0.119950
Score Fold 6 (Regularization=0.000083): 0.109066
Score Fold 7 (Regularization=0.000089): 0.113498
Score Fold 8 (Regularization=0.000094): 0.122939
Score Fold 9 (Regularization=0.000100): 0.117839
************************************************
*************** Model Parameters ****************
************************************************
Parameters of tokenizer:
TfidfVectorizer(max_df=0.3, max_features=5000, min_df=3, ngram_range=(1, 2),
                stop_words={´a´, ´able´, ´about´, ´across´, ´after´, ´all´,
                            ´almost´, ´also´, ´am´, ´among´, ´an´, ´and´, ´any´,
                            ´are´, ´as´, ´at´, ´be´, ´because´, ´been´, ´but´,
                            ´by´, ´can´, ´cannot´, ´could´, ´dear´, ´did´, ´do´,
                            ´does´, ´either´, ´else´, ...},
```

```
                sublinear_tf=True)
************************************************
Dimensions of document-n-gram-matrix:
(100000, 5000)
************************************************
Parameters of trained model:
Lasso(alpha=5.555555555555556e-05, random_state=5057)
************************************************
Chosen regularization strength:
5.555555555555556e-05
Model Score: 0.122109
************************************************
Step 4/4 :Saving Model

.
.
. **#* 2b)

. txtreg_train WOS_TOTAL_ln title using "$path/Models/predicted_citation_ridge.pkl"
>  , model("reg") regularization("ridge") regu_range(0.5,2) ngrams(2) seed(5184) tfi
> df stem stem_lang("english") stopwords("$stopwords") min_freq(3) max_freq(0.3) max
> _voc(5000)
  (output omitted)
.
.
. **#* 2c)

. txtreg_train WOS_TOTAL_ln title using "$path/Models/predicted_citation_elasticnet
> .pkl" , model("reg") regularization("elasticnet") regu_range(0.00005,0.0001) ngram
> s(2) seed(7469) tfidf stem stem_lang("english") stopwords("$stopwords") min_freq(3
> ) max_freq(0.3) max_voc(5000)
  (output omitted)

.
.
. **#* 2d)

. txtreg_train WOS_TOTAL_d title using "$path/Models/predicted_citation_logit_ridge
> .pkl" , model("logit") regularization("ridge") regu_range(1,10)  ngrams(2) seed(21
> 34) tfidf stem stem_lang("english") stopwords("$stopwords") min_freq(3) max_freq(0
> .3) max_voc(5000)
  (output omitted)
```

During the training process, `txtreg_train` reports the individual steps that were performed. In the first step, the data are loaded from Stata, and the command reports if the text strings are stemmed. The second step involves the creation of the document-n-gram-matrix. In each case, unigrams and bigrams are extracted from the text after the removal of stopwords and stemming.[4] Further, the document-n-gram-matrix is reweighted using tf-idf. In this way, `txtreg_train` automatically handles the preparation of the titles for the machine learning.

In step 3, `txtreg_train` reports the regularization strength that is used to train each fold and the resulting out-of-sample performance. If least squares regressions are used, the performance score is the out-of-sample R^2 of the text regression. In the case of logistic regressions, the out-of-sample F1-score is reported. At the end of the model training, the command reports the parameters of the vectorizer, which

4. To speed up the model training, the vocabulary is restricted to the 5,000 most frequent n-grams.

generates the document-n-gram-matrix and the parameters of the final model, including the chosen regularization strength and the out-of-sample performance score. For logistic regressions, the command also reports the confusion matrix (a matrix showing the actual vs. predicted outcomes). In the last step, the model is then saved in the specified location.

Making Use of Pre-trained Models To show the potential of the `txtreg_*` commands for data sharing, the author provides a text regression model which was pre-trained on over 24 Million papers in the fields of Medicine, Biology, Physics, Math and Chemistry for the years 1900-2010 based on the ISI Web of Science using a Ridge regression using the following line of code:[5]

```
. txtreg_train WOS_TOTAL_ln title using "$path/Models/predicted_citation_all.pkl"
>  , model("reg") regularization("ridge") regu_range("0.1,100") ngrams(2)
> seed(1502) tfidf stem stem_lang("english") stopwords("$stopwords") min_freq(100)
> max_freq(0.3) max_voc(200000)
```

The total training of this model took 2.5h. The model achieves an out-of-sample R^2 of 0.24. The `txtreg_predict` command can be used to load this model and generate a new variable containing the predicted citations. While the training of the model can be time and computationally intensive, the prediction is usually far quicker. In the example, the prediction took less than a minute, most of which is needed to stem the text strings.

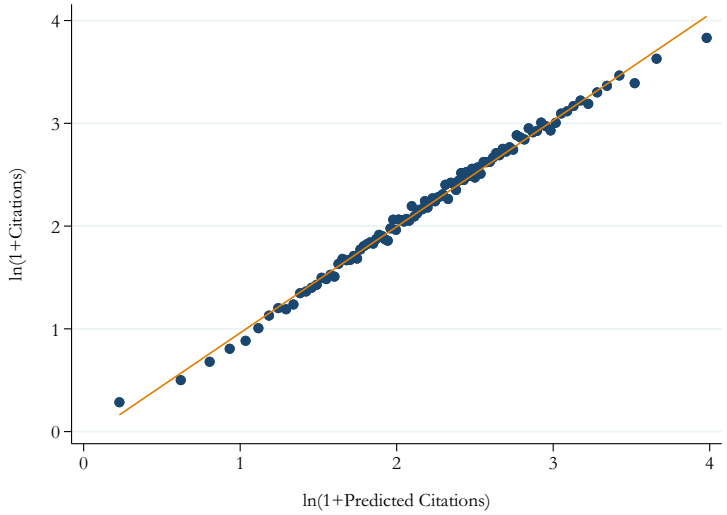
```
. ********************************************************************
. **#* 3) Predict outcome using txtreg_predict
. ********************************************************************
.
. txtreg_predict title using "$path/Models/predicted_citation_all.pkl" , name_new_v
> ar("predicted_citation") stem stem_Lang("english")
Loading Data from Stata
Loading Model: $path/Models/predicted_citation_all.pkl
Stemming text
```

After the command has run, the "predicted_citation" variable has been added to the data set. It is easy to show that the predicted citations are highly correlated with the actual citations using a binscatter plot (see Figure 1). The strong positive relationship suggests that the trained model performs very well. In the example, the original data already contained information on the citations. The pre-trained citation prediction model could alternatively be used to generate predicted citations for data sets where citations are either unknown or to create proxies of scientific influence for other text data sources.

Analyzing Model Coefficients Lastly, we can use the `txtreg_analyze` command to investigate if the model also identifies intuitive relationships between n-grams and citations, i.e., which n-grams have the largest predictive power for the outcome vari-

5. This pre-trained model is available on the author's website at www.carloschwarz.eu.

Figure 1: Predicted vs Actual Citations



Notes: This figure shows a binscatter of log(1+ Number of Predicted Citations) and the log(1+ Number of Citations). The number of citations is based on the ISI Web of Science.

able. This is achieved by specifying `txtreg_analyze` and the model path. Note that `txtreg_analyze` will replace the data set which is currently in memory. The new data set has one row for each n-gram in the vocabulary V and a variable containing the regression coefficient that is associated with each n-gram. It is then straightforward to obtain the most predictive n-grams by simply sorting the coefficients by size.

```
. ************************************************************************
. **#* 4) Analyze coefficients using txtreg_analyze
. ************************************************************************
.
. txtreg_analyze using "$path/Models/predicted_citation_all.pkl"
Loading Model: $path/Models/predicted_citation_all.pkl
  variable ngram was strL now str42
  (10,181,950 bytes saved)

.
.
. * list n-grams that are most predictive of high citations
. gsort -coef
. list ngram coef if _n<=10
```

	ngram	coef
1.	microrna	3.6021663
2.	graphen	3.5425722
3.	meta analysi	3.1363891
4.	random trial	3.1193305
5.	cut edge	3.0505556

6.	topolog insul	3.0193784
7.	organ framework	3.0116178
8.	arabidopsi	2.9907778
9.	energi harvest	2.9237338
10.	mice lack	2.8591244

The ten n-grams with the largest coefficients provide insights into which articles receive many citations. For example, the word meta-analysis appears. This makes intuitive sense as meta-analyses are often highly cited. Interestingly, also articles that use the n-gram "cutting edge" in their title receive more citations, as do articles using "randomized trials". Similarly, the n-grams "microRNA", "graphen", and "mice lack" all represent research topics of considerable interest. As the model was trained for all subjects and years combined, the ten most frequent words do, of course, not show the full scope of associations learned by the model. In Iaria et al. (2021) the predicted citation model is trained separately by cohort and subject, and the authors show that highly intuitive relationships emerge.

5 Conclusion

This paper described new commands that train text regression models in Stata. Aside from enabling quick training of text regressions, the `txtreg_*` commands provide researchers with the opportunity to share their pre-trained models. All models trained by `txtreg_train` can be loaded using the `txtreg_predict` and `txtreg_analyze` commands. The commands thereby further the use of text-based prediction across the community of Stata users.

6 References

Barker, M. 2012. STRDIST: Stata module to calculate the Levenshtein distance, or edit distance, between strings. *Statistical Software Components* .

Bird, S., E. Klein, and E. Loper. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. *" O'Reilly Media, Inc."* .

Gentzkow, M., B. Kelly, and M. Taddy. 2019. Text as data. *Journal of Economic Literature* 57(3): 535–74.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.

Iaria, A., C. Schwarz, and F. Waldinger. 2021. Gender Gaps in Academia: Global Evidence Over the Twentieth Century. *Working Paper* .

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel,

P. Prettenhofer, R. Weiss, V. Dubourg, et al.. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12(Oct): 2825–2830.

Schonlau, M., N. Guenther, and I. Sucholutsky. 2017. Text mining with n-gram variables. *The Stata Journal* 17(4): 866–881.

Schwarz, C. 2018. ldagibbs: A command for topic modeling in Stata using latent Dirichlet allocation. *Stata Journal* 18(1): 101–117.

———. 2019. lsemantica: A command for text similarity based on latent semantic analysis. *The Stata Journal* 19(1): 129–142.

The Pandas Development Team. 2020. Python Pandas. https://doi.org/10.5281/zenodo.3509134.

Williams, U., and S. P. Williams. 2014. txttool: Utilities for text analysis in Stata. *Stata Journal* 14(4): 817–829.