SAS

Natural Language Processing with SAS®

Special Collection

Foreword by
Katie Tedrow

The correct bibliographic citation for this manual is as follows: Tedrow, Katie. *Natural Language Processing with SAS®: Special Collection*. Cary, NC: SAS Institute Inc.

Natural Language Processing with SAS®: Special Collection

Copyright © 2020, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-952363-18-4 (Paperback)
ISBN 978-1-952363-16-0 (Web PDF)

All Rights Reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

August 2020

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SAS software may be provided with certain third-party software, including but not limited to open-source software, which is licensed under its applicable third-party software license agreement. For license information about third-party software distributed with SAS software, refer to **http://support.sas.com/thirdpartylicenses**.

Table of Contents

Free SAS® e-Books: Special Collection

In this series, we have carefully curated a collection of papers that introduces and provides context to the various areas of analytics. Topics covered illustrate the power of SAS solutions that are available as tools for data analysis, highlighting a variety of commonly used techniques.

Discover more free SAS e-books!
support.sas.com/freesasebooks

sas.com/books
for additional books and resources.

THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2017 SAS Institute Inc. All rights reserved. M1673525 US.0817

Foreword

Did you know that unstructured text is the largest human-generated data source? Each day worldwide, we send on average more than 500 million Tweets, about 5.5 billion SMS text messages, and over 281 billion emails.

Tons of unstructured data is collected within organizations every day, too – from customer call logs, emails, social media, surveys, product feedback, documents and reports, and so on. Often buried within that unstructured data are rich insights that can help drive better business decisions, inform product strategy, and improve customer experiences.

This is where Natural Language Processing (NLP) comes in. NLP is an umbrella term used to describe a branch of artificial intelligence that helps computers understand, interpret, and emulate written or spoken human language. NLP draws from many disciplines including human-generated linguistic rules, machine learning, and deep learning to fill the gap between human communication and machine understanding.

NLP can be used to scale the human act of reading, organizing, and quantifying text data. Taking it a step further, NLP can empower conversational experiences, where a machine actually understands and responds and interacts with a user through natural language (often referred to as conversational AI). While this may sound futuristic, applying NLP to solve real business problems is becoming pervasive across industries, and new advancements are continuing to extend possibilities of what we can do with unstructured data.

Several groundbreaking papers have been written to demonstrate these techniques and practical applications. We have carefully selected a handful of these from recent SAS Global Forum papers to introduce you to the topics and let you sample what each has to offer. You will learn about the following:

- how NLP helps analysts combat anti-money laundering and fraud in the financial services industry.
- how NLP can help governments and policy makers in the rulemaking process by analyzing thousands of comments on proposed rules much faster and more accurately than would be possible manually.
- how to approach multilingual sentiment analysis when limited data is available.
- how to build text analytics models in SAS using open source, such as Python, to analyze product reviews.
- how to create your own BERT sentiment analysis model using SAS.
- how to extract hidden value from audio data using speech to text and machine learning to mine key voice of customer insights.

For an in-depth, technical understanding of text analytics and how to get started, I recommend reading *SAS® Text Analytics for Business Applications: Concept Rules for Information Extraction* by Teresa Jade, Biljana Belamaric-Wilsey, and Michael Wallis.

The examples found in this e-book are from across industries and offer various business use cases to share a glimpse into how text analytics can be applied in your organizations. The beauty of NLP is that it has wide application – wherever there may be unstructured data, NLP can help discover opportunities and insights to drive decisions.

How to Build a Text Analytics Model in SAS® Viya® with Python

By Nate Gilmore, Vinay Ashokkumar, and Russell Albright

Python is widely noted as one of the most important languages influencing the development of machine learning and artificial intelligence. SAS has made seamless integration with Python one of its recent focal points. With the introduction of the SAS Scripting Wrapper for Analytics Transfer (SWAT) package, Python users can now easily take advantage of the power of SAS Viya. This paper is designed for Python users who want to learn more about getting started with SAS Cloud Analytic Services (CAS) actions for text analytics. It walks them through the process of building a text analytics model from end to end by using a Jupyter Notebook as the Python client to connect to SAS Viya. Areas that are covered include loading data into CAS, manipulating CAS tables by using Python libraries, text parsing, converting unstructured text into input variables used in a predictive model, and scoring models. The ease of use of SWAT to interact with SAS Viya using Python is showcased throughout the text analytics model building process.

Harvesting Unstructured Data to Reduce Anti-Money Laundering (AML) Compliance Risk

By Austin Cook and Beth Herron

As an anti-money laundering (AML) analyst, you face a never-ending job of staying one step ahead of nefarious actors (for example, terrorist organizations, drug cartels, and other money launderers). The financial services industry has called into question whether traditional methods of combating money laundering and terrorism financing are effective and sustainable. Heightened regulatory expectations, emphasis on 100% coverage, identification of emerging risks, and rising staffing costs are driving institutions to modernize their systems. One area gaining traction in the industry is to leverage the vast amounts of unstructured data to gain deeper insights. From suspicious activity reports (SARs) to case notes and wire messages, most financial institutions have yet to apply analytics to this data to uncover new patterns and trends that might not surface themselves in traditional structured data. This paper explores the potential use cases for text analytics in AML and provides examples of entity and fact extraction and document categorization of unstructured data using SAS® Visual Text Analytics.

Hearing Every Voice: SAS® Text Analytics for Federal Regulations Public Commentary

By Emily McRae, Tom Sabo, and Manuel Figallo

Regulations.gov was launched in 2003 to provide the public with access to federal regulatory content and the ability to submit comments on federal regulations. Public participation in federal rulemaking is encouraged as it supports the legitimacy of regulatory decisions, frames public acceptance or resistance to rules under development, and shapes how the public interest will be served. Manually reading thousands of comments is time-consuming and labor-intensive. It is also difficult for multiple reviewers to accurately and consistently assess content, themes, stakeholder identity, and sentiment. Given that individually proposed rules can exceed 10,000 comments, how can federal organizations quantitatively assess the data and incorporate feedback into the rulemaking process as required by law?

This paper shows how SAS® Text Analytics can be used to develop transparent and accurate text models, and how SAS® Visual Analytics can quantify, summarize and present the results of that analysis. This will significantly decrease time to value, leveraging capabilities that computers excel at while freeing up human intuition for the analysis of these results. Specifically, we will address public commentary submitted in response to new product regulations by the US Food and Drug Administration. Ultimately, the application of a transparent and consistent text model to analyze these documents will support federal rule-makers and improve the health and lives of American citizens.

Multilingual Sentiment Analysis: An RNN-Based Framework for Limited Data

By Ethem Can and Aysu Ezen-Can

Sentiment analysis is a widely studied natural language processing task, whose goal is to determine users' opinions, emotions, and evaluations of a product, entity, or service that they review. One of the biggest challenges for sentiment analysis is that it is highly language-dependent. Word embeddings, sentiment lexicons, and even annotated data are language-specific. Furthermore, optimizing models for each language is very time-consuming and labor-intensive, especially for recurrent neural network (RNN) models. From a resource perspective, it is very challenging to collect data for different languages.

In this paper, we look for an answer to the following research question: Can a sentiment analysis model that is trained on one language be reused for sentiment analysis in other languages where the data are more limited? Our goal is to build a single model in the language that has the largest data set available for the task and reuse that model for languages that have limited resources.

For this purpose, we use reviews in English to train a sentiment analysis model by using recurrent neural networks. We then translate those reviews into other languages and reuse the model to evaluate the sentiments. Experimental results show that our robust approach of training a single model on English-language reviews outperforms the baseline in several different languages.

NLP with BERT: Sentiment Analysis Using SAS® Deep Learning and DLPy

By Doug Cairns and Xiangxiang Meng

A revolution is taking place in natural language processing (NLP) as a result of two ideas.

The first idea is that pretraining a deep neural network as a language model is a good starting point for a range of NLP tasks. These networks can be augmented (layers can be added or dropped) and then fine-tuned with transfer learning for specific NLP tasks. The second idea involves a paradigm shift away from traditional recurrent neural networks (RNNs) and toward deep neural networks based on Transformer building blocks. One architecture that embodies these ideas is Bidirectional Encoder Representations from Transformers (BERT). BERT and its variants have been at or near the top of the leaderboard for many traditional NLP tasks, such as the general language understanding evaluation (GLUE) benchmarks. This paper provides an overview of BERT and shows how you can create your own BERT model by using SAS® Deep Learning and the SAS DLPy Python package. It illustrates the effectiveness of BERT by performing sentiment analysis on unstructured product reviews submitted to Amazon.

Sound Insights: A Pipeline for Information Extraction from Audio Files

Dr. Biljana Belamarić Wilsey and Xiaozhuo Cheng

Audio files, like other unstructured data, present special challenges for analytics but also an opportunity to discover valuable new insights. For example, technical support or call center recordings can be used for quickly prioritizing product or service improvements based on the voice of the customer. Similarly, audio portions of video recordings can be mined for common topics and widespread concerns. To uncover the value hidden in audio files, you can use a pipeline that starts with the speech-to-text capabilities of SAS® Visual Data Mining and Machine Learning and continues with analysis of unstructured text using SAS® Visual Text Analytics software. This pipeline can be illustrated with data from the Big Ideas talk series at SAS, which gives employees the opportunity to share their ideas in short, TED Talk–type presentations that are recorded on video. If you ever wondered what SAS employees are thinking about when they're not thinking of ways to make SAS products better, the answers lie in a pipeline for information extraction from audio files. You can use this versatile pipeline to discover sound insights from your own audio data.

We hope these selections give you a useful overview of the many tools and techniques that are available to incorporate analysis of unstructured text into your organization.

Katie Tedrow is a Global Product Marketing Manager for AI at SAS. In her role, she leads product marketing for Natural Language Processing with a specialization in text analytics, conversational AI, and chatbots, in addition to Business Intelligence and Visual Analytics. Katie is a strategic marketer with deep B2B and B2C experience within the professional services, tech, and financial services industries. Prior to joining SAS, Katie was a product marketing and digital brand strategy lead at a large financial services company, where she helped to launch the first natural language chatbot for a US bank. She holds a bachelor's degree from North Carolina State University and an MBA from the University of Maryland.

Paper SAS4442-2020

How to Build a Text Analytics Model in SAS® Viya® with Python

Nate Gilmore, Vinay Ashokkumar, and Russell Albright, SAS Institute Inc.

ABSTRACT

Python is widely noted as one of the most important languages influencing the development of machine learning and artificial intelligence. SAS® has made seamless integration with Python one of its recent focal points. With the introduction of the SAS® Scripting Wrapper for Analytics Transfer (SWAT) package, Python users can now easily take advantage of the power of SAS® Viya®. This paper is designed for Python users who want to learn more about getting started with SAS® Cloud Analytic Services (CAS) actions for text analytics. It walks them through the process of building a text analytics model from end to end by using a Jupyter Notebook as the Python client to connect to SAS Viya. Areas that are covered include loading data into CAS, manipulating CAS tables by using Python libraries, text parsing, converting unstructured text into input variables used in a predictive model, and scoring models. The ease of use of SWAT to interact with SAS Viya using Python is showcased throughout the text analytics model building process.

INTRODUCTION

One of the ways that SAS provides access to its high-quality text analytics services is through CAS actions that can be invoked directly from SAS, Python, Lua, or R. These actions cover a wide array of functionality including text mining, text categorization, concept identification, and sentiment analysis.

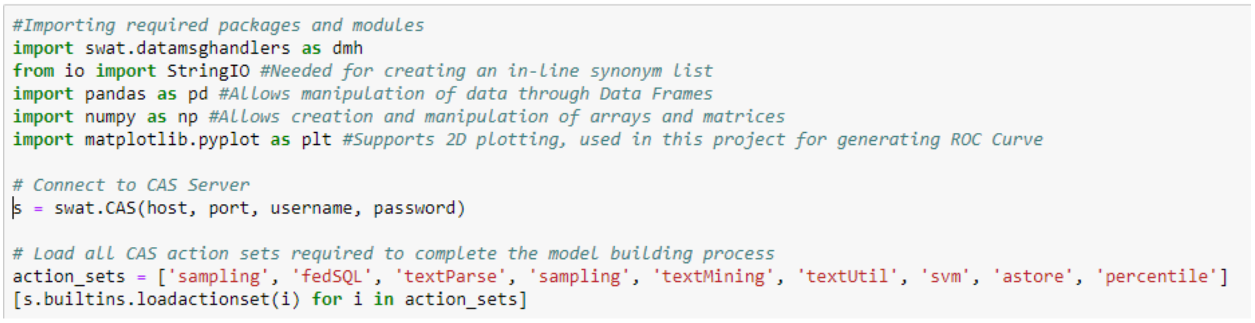
This paper highlights the construction of an end-to-end text analytics model that leverages CAS actions called from Python. It demonstrates how the unstructured text of Amazon reviews can be converted into structured input variables and used in a Support Vector Machine (SVM) model to predict whether users will find an Amazon review helpful. The client-side platform used is a Jupyter notebook, a very popular interface for Python users. The version of Python used for this project was Python 3.4.1.

PYTHON AND CAS

In recent history, many users have embraced Python as a first-choice programming language for the development of software across all domains. The community has embraced Python's open source nature and ease of use through many functional libraries to aid in design and performance. As such, the Python user space continues to grow unfailingly.

SAS Viya has opened its arms to Python users by allowing integration of open source to its platform and its use of features. SAS has introduced SWAT (Scripting Wrapper for Analytics Transfer), a Python library that enables users, even those with no SAS background, the ability to continue coding in Python, while leveraging the performance and resources of CAS and SAS Viya in their applications. Users are able to create, manipulate, and print data with CAS actions through the Python interface. Added performance improvements are due to the data being processed while on the cloud as a part of the SAS Viya architecture and the data being loaded and worked on in memory. Other popular libraries, like pandas and NumPy, are supported to allow increased compatibility with the open source tools.

Figure 1 below shows the Python libraries that are imported and used in this project. Under that, the basic syntax for connecting to CAS through SWAT is shown, including using the user's credentials. The final lines of code define and load all the action sets used in this project as a list. The relevant CAS actions under each respective action set will be detailed in the upcoming sections.



```python
#Importing required packages and modules
import swat.datamsghandlers as dmh
from io import StringIO #Needed for creating an in-line synonym list
import pandas as pd #Allows manipulation of data through Data Frames
import numpy as np #Allows creation and manipulation of arrays and matrices
import matplotlib.pyplot as plt #Supports 2D plotting, used in this project for generating ROC Curve

# Connect to CAS Server
s = swat.CAS(host, port, username, password)

# Load all CAS action sets required to complete the model building process
action_sets = ['sampling', 'fedSQL', 'textParse', 'sampling', 'textMining', 'textUtil', 'svm', 'astore', 'percentile']
[s.builtins.loadactionset(i) for i in action_sets]
```

Figure 1. Importing Libraries, Connecting to CAS, and Loading Action Sets

USE CASE – PREDICTING REVIEW HELPFULNESS

The data set analyzed in this paper includes over 67,000 Amazon reviews of fine food products. The model built in this project will predict whether a review will be rated helpful or not by Amazon users. For the purposes of this paper, a helpful review is defined as one that at least 80% of voters found helpful (with a minimum of 5 users having voted on its helpfulness). The explanatory variables considered include the star rating of the review, the length of the review, and the text of the reviews. The text analytics portion of the model building process focuses on converting the unstructured text of the review into document projections that will be used as input variables to the SVM predictive model along with the star rating and review length. The Analyzing Results section of this paper details how including document projections derived from the review text significantly improves the predictive model's accuracy compared to using only star rating and review length. Table 1 below describes the most relevant variables in the data set, which will be referenced in the code snippets of upcoming sections.

Variable	Description
ID	A unique identifier for each review
Text	Text of the product review
Score	Star rating for a review (From 1 to 5)
Review_Length	Number of words in a review
Helpful	Target Variable (1=Helpful, 0=Not Helpful)

Table 1. Description of Relevant Variables

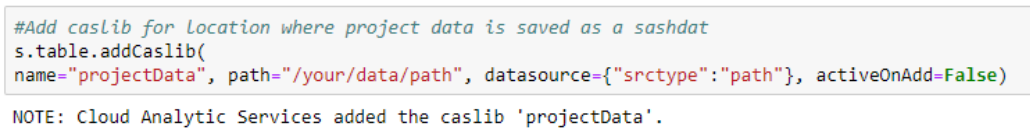
LOADING DATA INTO CAS

The data preprocessing steps necessary to prepare the data for consumption by the model were performed in Python on the client side prior to loading data into CAS. Those steps included:

1. Dropping all observations that did not have at least five helpfulness ratings.
2. Creating a predictor variable containing the number of words in a review.

3. Creating a unique identifier variable for each review as required by the text actions.
4. Defining a target variable for review helpfulness as indicated in the previous section.

After preprocessing, the first step in preparing the data to be loaded into CAS is to define a casLib to the location where your data is stored. Figure 2 shows how to use the addCaslib action to define a caslib named "projectData " in the location where the input data is stored.

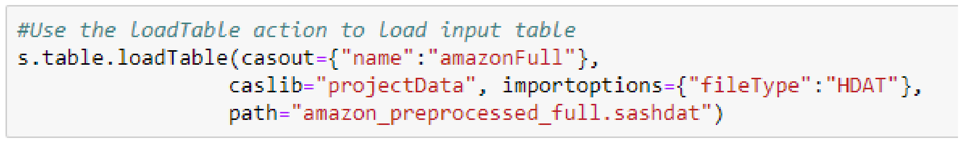


```
#Add caslib for location where project data is saved as a sashdat
s.table.addCaslib(
name="projectData", path="/your/data/path", datasource={"srctype":"path"}, activeOnAdd=False)
```

```
NOTE: Cloud Analytic Services added the caslib 'projectData'.
```

Figure 2. addCaslib Action Code

After setting up the caslib, the loadTable action makes it easy to load your data to CAS. Figure 3 shows how to load the preprocessed Amazon Fine Food Reviews data set, stored as a .sashdat, into CAS. The data is stored as a CAS table named "amazonFull".

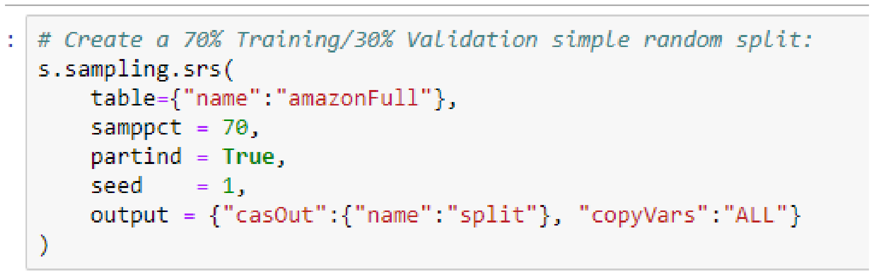


```
#Use the loadTable action to load input table
s.table.loadTable(casout={"name":"amazonFull"},
                  caslib="projectData", importoptions={"fileType":"HDAT"},
                  path="amazon_preprocessed_full.sashdat")
```

Figure 3. Loading Amazon Reviews into CAS with the loadTable Action

SPLITTING DATA INTO TRAINING AND VALIDATION SETS

Now that the model's input data has been loaded into CAS, the next step is to split the data into training and validation sets. For this project, 70% of reviews were included in the training set while 30% were reserved for validation. Reviews were assigned to either the training or validation data set via simple random sample using the srs CAS action as shown in Figure 4.



```
: # Create a 70% Training/30% Validation simple random split:
  s.sampling.srs(
      table={"name":"amazonFull"},
      samppct = 70,
      partind = True,
      seed    = 1,
      output = {"casOut":{"name":"split"}, "copyVars":"ALL"}
  )
```

Figure 4. Using the SRS Action to Create 70% Training/30% Validation

The srs action assigns a partition index, *_PartInd_*, which takes a value of 1 for reviews assigned to the training set and 0 for reviews assigned to the validation set. The execDirect action from the fedSQL action set is used to create separate CAS tables for the training and validation sets as shown in Figure 5.

```
#Splitting training and validation sets into separate tables
s.fedSQL.execDirect('''
  CREATE TABLE "amazonTraining" AS SELECT * FROM SPLIT WHERE "_PartInd_" = 1
''')
s.fedSQL.execDirect('''
  CREATE TABLE "amazonValidation" AS SELECT * FROM SPLIT WHERE "_PartInd_" = 0
''')
```

Figure 5. Using the execDirect Action to Split Training and Validation into Separate CAS Tables

DATA EXPLORATION

The next step in the model building process is to explore your training data. Initial data exploration shows that 63.7% of the training data set is comprised of helpful reviews while the remaining 36.3% of reviews are unhelpful. Further exploration shows that 50% of the reviews were rated 5 stars, 24% are rated 1 star, and the remaining 26% are split rather evenly between 2, 3, and 4 stars. One noteworthy finding is that while 86.2% of 5-star reviews are considered helpful, only 27.5% of 1-star reviews are considered helpful. This exploration process gives you an idea that the star rating will likely be a very useful variable in determining the likelihood that a review is helpful. Users are more likely to consider reviews with higher star ratings as helpful than those that have lower star ratings. Figure 6 shows the code to perform a cross-tabulation using the pandas library along with the resulting stacked bar chart that demonstrates how the proportion of reviews that are considered helpful varies for each level of review rating.

```
#Converting CAS Table into CAS Table Object (Intermediate data structure required prior to converting to data frame)
training = s.CASTable("amazonTraining")

#Converting the training data to a data frame
df_training = training.to_frame()

#Performing a cross-tabulation of Star Rating and Review Helpfulness
tabulation = pd.crosstab(df_training.score, df_training.helpful)

#Add column Headers
tabulation.columns = ['Unhelpful Total', 'Helpful Total']

#Creating Stacked Bar Chart
tf = tabulation.plot.bar(stacked=True, title="Review Helpfulness by Review Rating")
tf.set_xlabel("User Rating (Stars)")
tf.set_ylabel("Number of Observations")
```

```
Text(0,0.5,'Number of Observations')
```

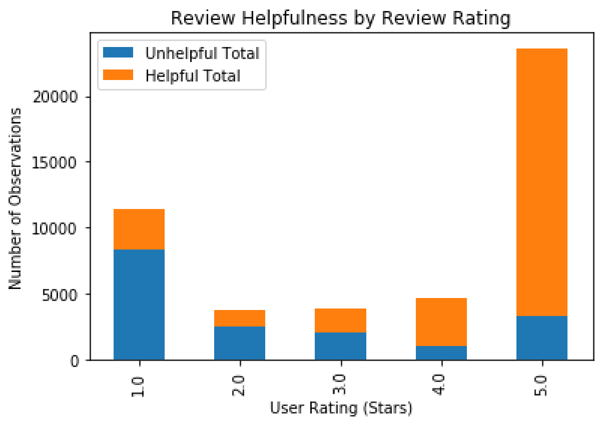


Figure 6. Relationship between Review Rating and Review Helpfulness

MODEL TRAINING

This section shows the various steps in building a predictive model that includes text. The code not only includes a standard predictive model, in this case we use an SVM, but also the actions needed to transform your unstructured data to a numeric representation.

Figure 7 shows the general overview of the code needed to create the models for scoring. There are two major components, one for text and the other for the predictive model. Both sections produce their own analytic store model. In the following subsections, these model training components are covered more specifically.

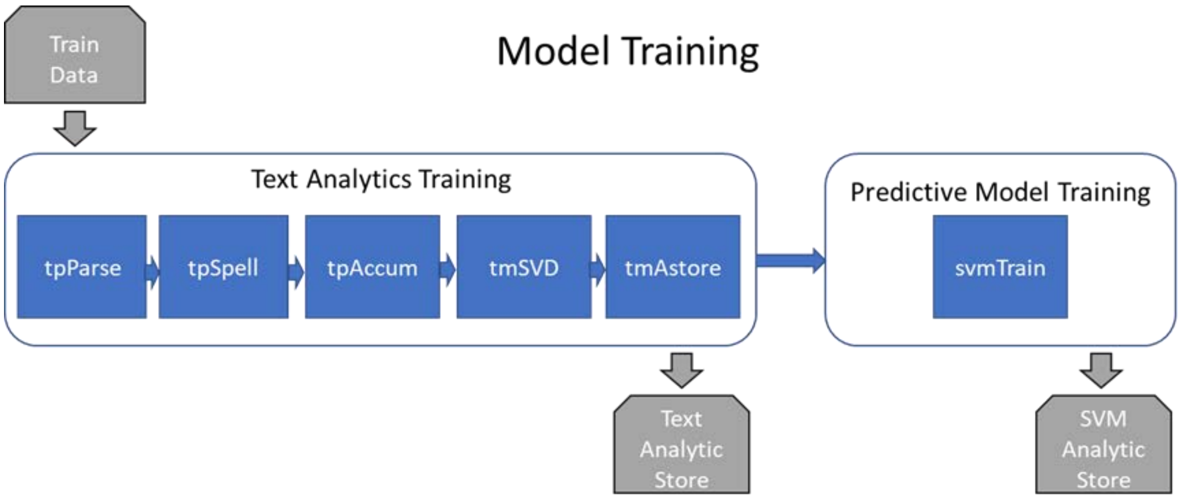


Figure 7. Model Training Overview

PARSING THE TEXT

The tpParse action parses each row of the input table and creates an output offset table that lists every term found in every document. It is the first step in transforming your text to a numerical representation. There are many options to control how the tokenization works and to enable you to use various natural language features such as the part-of-speech tags or the stemmed form of a term. In the code shown in Figure 8, you can see common settings used in the tpParse action. You should explore which settings work best for your particular input data and subsequent model.

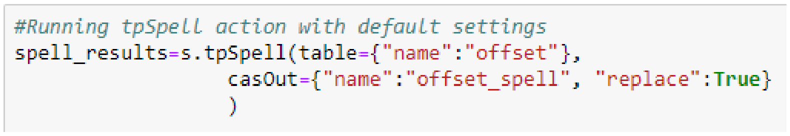
```
#Call tpParse action to parse the text of each review
parsed_results=s.tpParse(table={"name":"amazonTraining"},
                   docid="id",
                   text="text",
                   entities="std",
                   noungroups=True,
                   stemming=True,
                   tagging=True,
                   outComplexTag=True,
                   predefinedMultiterm=True,
                   language="English",
                   offset={"name":"offset", "replace":True},
                   parseConfig={"name":"outConfig", "replace":True}
                   )
```

Figure 8. The tpParse Action Code

In addition to the output offset table, you should also request the parseConfig output. This table stores the settings that you use so that they can be reused at score time. Below, you will see that the parseConfig table is added to and then used to build a scoring model.

CORRECTING MISSPELLINGS

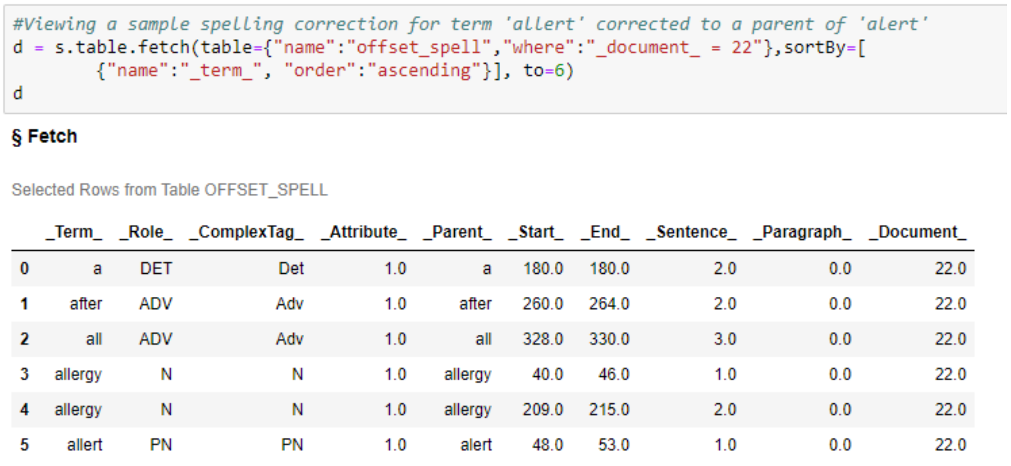
When your input text data is particularly noisy, such as informal chat messages or other unedited content, the tpSpell action can be useful for automatically correcting misspellings. This action takes the offset table from the tpParse action, analyzes it for spelling corrections that need to be made, and, on output, updates the offset table with these corrections. The tpSpell action finds candidate misspellings by looking across the entire collection for rare terms that are very similar in spelling to more common terms. The code for calling the tpSpell action is shown in Figure 9.



```
#Running tpSpell action with default settings
spell_results=s.tpSpell(table={"name":"offset"},
                        casOut={"name":"offset_spell", "replace":True}
                        )
```

Figure 9. The tpSpell Action Code

In Figure 10, you can see the fetch action that retrieves and displays a subset of the output table from the tpSpell action. This output table replaces the offset table of tpParse, correcting the parent values of misspelled terms. In the table shown in Figure 10, the misspelled term "allert" has been corrected to having a parent of "alert".



```
#Viewing a sample spelling correction for term 'allert' corrected to a parent of 'alert'
d = s.table.fetch(table={"name":"offset_spell","where":"_document_ = 22"},sortBy=[
        {"name":"_term_", "order":"ascending"}], to=6)
d
```

§ Fetch

Selected Rows from Table OFFSET_SPELL

	Term	_Role_	_ComplexTag_	_Attribute_	_Parent_	_Start_	_End_	_Sentence_	_Paragraph_	_Document_
0	a	DET	Det	1.0	a	180.0	180.0	2.0	0.0	22.0
1	after	ADV	Adv	1.0	after	260.0	264.0	2.0	0.0	22.0
2	all	ADV	Adv	1.0	all	328.0	330.0	3.0	0.0	22.0
3	allergy	N	N	1.0	allergy	40.0	46.0	1.0	0.0	22.0
4	allergy	N	N	1.0	allergy	209.0	215.0	2.0	0.0	22.0
5	allert	PN	PN	1.0	alert	48.0	53.0	1.0	0.0	22.0

Figure 10. The Fetch Action Code Producing the Output from tpSpell

GENERATING A TERM-BY-DOCUMENT MATRIX

Once the text has been tokenized into the offset table, you use the tpAccumulate action to filter and reassign some of the terms, and to create a term-by-document weighted frequency table.

Filtering and reassigning the terms is done with the following options on the tpAccumlulate action:

- synonyms: Maps a set of terms to a canonical form of those terms and reduces the number of terms in your analysis.
- stopList: Eliminates specific terms from your analysis. A default stopList is provided in the reference library.
- reduce: Throws out infrequently occurring terms as these tend to be just noise in the collection.

In Figure 11 below, you can see how to create a custom synonym list to use as input to the tpAccumulate action using Python's StringIO function and SWAT's data message handler.

```
#Create Synonym List using Python's StringIO function
synonyms = StringIO('''term, termrole, parent, parentrole
"tsp", "N", "teaspoon", "N"
"tbsp", "N", "tablespoon", "N"
"carb", "N", "carbohydrate", "N"
"fridge", "N", "refrigerator", "N"
"hot chocolate", "nlpNounGroup", "hot cocoa", "nlpNounGroup"
"purchase", "V", "buy", "V"
"tummy", "N", "stomach", "N"
"tasty", "A", "delicious", "A"
"yummy", "A", "delicious", "A"
"yucky", "A", "disgusting", "A"
"mom", "N", "mother", "N"
"dad", "N", "father", "N"
"begin", "V", "start", "V"
"fast", "A", "quick", "A" ''')

#Add a handler to transport and load data into CAS using addTable action
handler = dmh.CSV(synonyms, skipinitialspace=True)
s.addtable(table='synonyms', replace=True, **handler.args.addtable)
```

Figure 11. An Example of Creating a Synonym List

Figure 12 shows the call to the tpAccumulate action. For your particular problem, you should consider experimenting with the different termWeight settings, the reduce= setting, and modify the terms on your stop and synonym lists to be useful for your data. Often the Mutual Information weighting, in conjunction with a target input is helpful, but in this case the setting seemed to cause overfitting, so it was not used.

```
#Running tpAccumulate action to generate term by document matrix
accumulate=s.tpAccumulate(cellWeight="LOG",
                           termWeight="entropy",
                           reduce=10,
                           offset={"name":"offset_spell"},
                           stopList={"name":"stop_list"},
                           synonyms={"name":"synonyms"},
                           terms={"name":"outterms", "replace":True},
                           parent={"name":"outparent", "replace":True},
                           complexTag=True)
```

Figure 12. The tpAccumulate Action Code

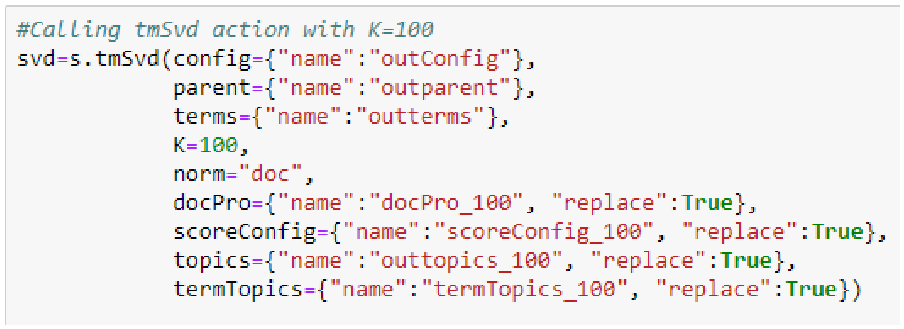
There are two primary outputs of the tpAccumulate action. The first is the terms table, which is a summary table containing the unique terms in the collection and the frequencies at which they occur. The second is the parent table, which is a compressed representation of the term-by-document weighted frequency table.

GENERATING DOCUMENT PROJECTIONS

In a term-by-document weighted frequency matrix, each document is represented with a vector whose length is equal to the number of distinct terms in the collection. While this is a numerical representation, it is too long and sparse to be useful, so your transformation of your input text to a numerical representation will be complete when the term-by-document frequency matrix is projected onto a smaller dimensional space. The tmSVD action in the textMining action set enables you to form this projection.

The action can do much more, such as discover topics in your data, but for your predictive model, you are primarily interested in the docPro table containing the k real-valued _Col1_-_Colk_ variables, where k is the number of dimensions you choose. These document projections variables, in conjunction with any other variables on your training data that you think might be useful, can be used as input when you train your predictive model.

The tmSVD action call shown in Figure 13 has several output tables. In addition to the docpPro table, the output scoreConfig table is the same parseConfig table you created with tpParse, together with additional information that tmSVD model needs at score time. The topics and termtopics output tables are not specifically required for the predictive model, but they are required for making the analytic store in the next subsection, so they are also requested. The option norm=″doc″ is specified to override the optimal topic calculation and instead focus on getting the best predictive ability. When you set the norm option in this way, you make sure that the document projections are normalized to unit length.



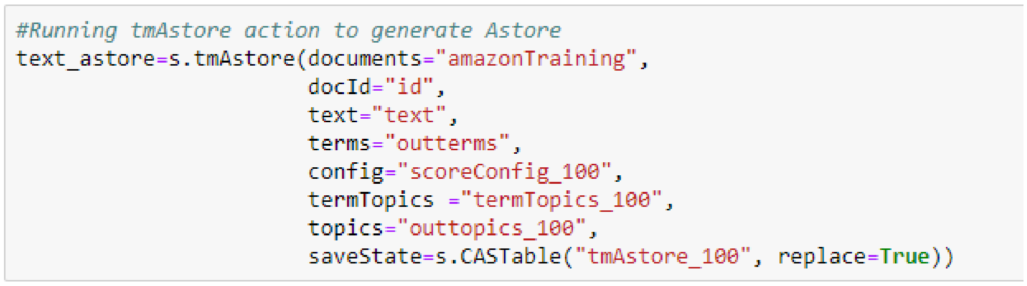
```
#Calling tmSvd action with K=100
svd=s.tmSvd(config={"name":"outConfig"},
            parent={"name":"outparent"},
            terms={"name":"outterms"},
            K=100,
            norm="doc",
            docPro={"name":"docPro_100", "replace":True},
            scoreConfig={"name":"scoreConfig_100", "replace":True},
            topics={"name":"outtopics_100", "replace":True},
            termTopics={"name":"termTopics_100", "replace":True})
```

Figure 13: The tmSvd Action Code

CREATING A TEXT MINING ANALYTIC STORE

The analytic store scoring mechanism has become a standard across SAS. The approach encapsulates needed information and data into a binary object, which is used for model deployment. In the tmAstore action, you create an analytic store by combining the content of several tables that were generated from the previous actions into a single analytic store table object. This table will be applied at score time with the score action from the analytic store action set.

Figure 14 contains the code to create the output analytic store table for the text analytics portion of your predictive model. In the following section you will see how to create a second analytic store for the SVM portion of your model.



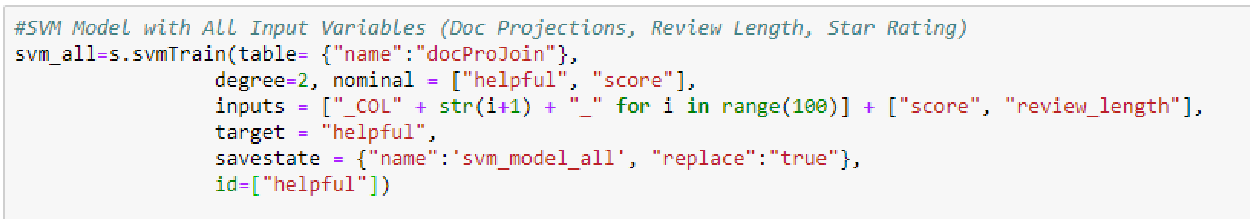
```
#Running tmAstore action to generate Astore
text_astore=s.tmAstore(documents="amazonTraining",
                       docId="id",
                       text="text",
                       terms="outterms",
                       config="scoreConfig_100",
                       termTopics ="termTopics_100",
                       topics="outtopics_100",
                       saveState=s.CASTable("tmAstore_100", replace=True))
```

Figure 14. The tmAstore Action Code

BUILDING PREDICTIVE SUPPORT VECTOR MACHINES MODEL

The step above describes the final step of the text analytics training stages. The following stages defined from here are the predictive training stages. In this section, details on the analytical actions used for building the machine learning model will be described. As stated in the introduction, the model chosen is SVM (Support Vector Machine). An SVM is a type of classifier that can learn from labeled training data and identify features that distinguish between different classes. For this project, the two classes that the SVM will classify are between reviews considered helpful and reviews considered unhelpful. To aid in the classification using an SVM, there are certain mathematical functions that are supported called kernels. Kernels are used to transform the data into a form that can make the data easier to classify. The purpose of these functions is to improve the separability in the data. Separability, in terms of SVM classification, is a property where two or more sets of data points can be easily divided into different classes. The functions transform the points so that a linear separator can be found that will make fewer errors.

Figure 15 below demonstrates the use of the svmTrain action, which uses linear and non-linear kernels to compute support vector machine (SVM) learning classifiers for the binary pattern recognition problem. The "degree" parameter specifies the degree of the polynomial kernel used. The "input" parameters specify the explanatory variables used for analysis. The document projections generated from the previous steps, as well as the review rating and review length are selected as these variables. The target variable, "helpful" is specified in the "target" parameter. When the action is run, the training results can be saved to a CAS table containing an analytic store and specified with the savestate option.



```
#SVM Model with All Input Variables (Doc Projections, Review Length, Star Rating)
svm_all=s.svmTrain(table= {"name":"docProJoin"},
                   degree=2, nominal = ["helpful", "score"],
                   inputs = ["_COL" + str(i+1) + "_" for i in range(100)] + ["score", "review_length"],
                   target = "helpful",
                   savestate = {"name":'svm_model_all', "replace":"true"},
                   id=["helpful"])
```

Figure 15. The svmTrain Action Code

MODEL SCORING

In this section you will see how to apply an analytic store scoring model. This standardized approach makes model deployment an easy step in many different contexts. Analytic stores are designed so that multiple different analytic stores can be applied one after another. In this case, you will first use the analytic store created from the text analytics component and then you will use the analytic store that encapsulates the SVM model. Figure 16 illustrates the model scoring process.

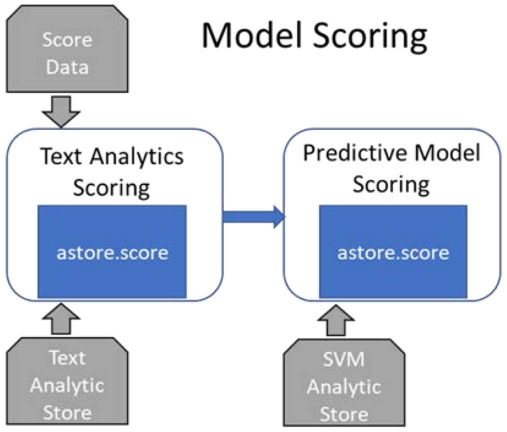
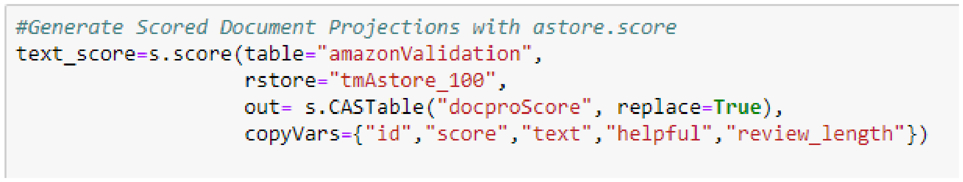


Figure 16. The Model Scoring Process

GENERATING SCORED DOCUMENT PROJECTIONS

You apply the text analytics scoring on the validation data with the score action. Your two inputs are the data you want to score and the text analytic store table. If there are additional variables that you want to use, then add them with the copyVars statement and they will be passed along together with the _Col1_-_Colk_ variables that you created at train time. The score action call is shown in Figure 17 and the output table created with the out option will then serve as input to the SVM scoring.

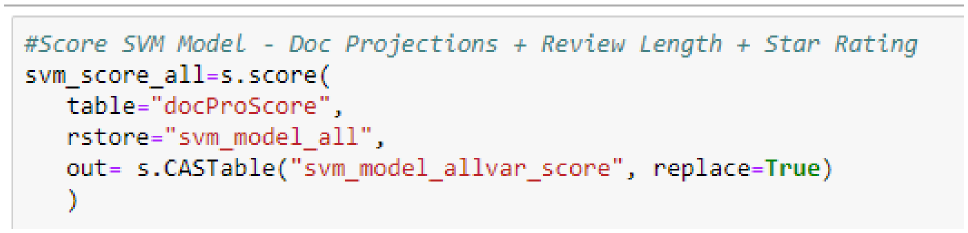


```
#Generate Scored Document Projections with astore.score
text_score=s.score(table="amazonValidation",
                   rstore="tmAstore_100",
                   out= s.CASTable("docproScore", replace=True),
                   copyVars={"id","score","text","helpful","review_length"})
```

Figure 17. Analytic Store Scoring Code for Text Model

SCORING THE SUPPORT VECTOR MACHINES PREDICTIVE MODEL

To score the SVM model, the output from the text scoring step and the SVM analytic store are submitted as input into the score action. The score action code for scoring the SVM model is shown in Figure 18. Your predictions will be included in the resulting output table.

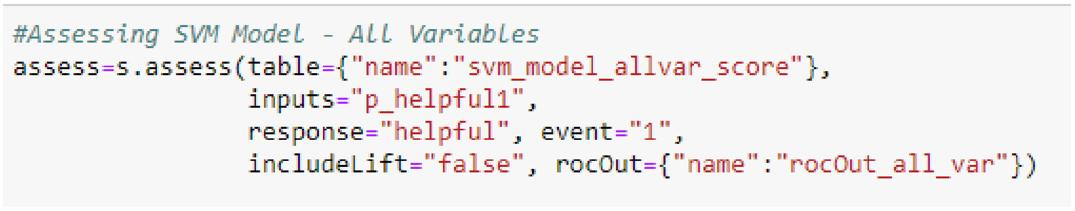


```
#Score SVM Model - Doc Projections + Review Length + Star Rating
svm_score_all=s.score(
   table="docProScore",
   rstore="svm_model_all",
   out= s.CASTable("svm_model_allvar_score", replace=True)
   )
```

Figure 18. Analytic Store Scoring Code for SVM Model

ASSESSING RESULTS

To assess the results of your scored SVM model, you can use the assess action from the percentile action set. This action generates an output table with Receiver Operating Characteristic (ROC) information. Included in this output is the C statistic, which represents the area under the ROC curve, a common metric for assessing model performance. Figure 19 shows the code to generate the ROC information for your scored model.



```
#Assessing SVM Model - All Variables
assess=s.assess(table={"name":"svm_model_allvar_score"},
                inputs="p_helpful1",
                response="helpful", event="1",
                includeLift="false", rocOut={"name":"rocOut_all_var"})
```

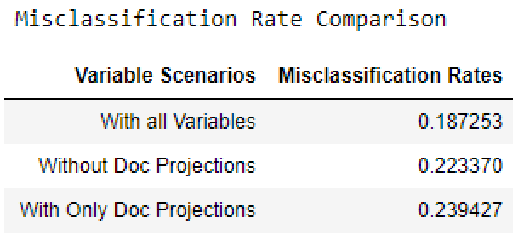
Figure 19. Assess Action Code to Generate ROC Table

From the ROC output table, you will find that the area under the ROC curve for your model is 0.86 and that the overall misclassification at a p=0.50 cutoff is 18.7%. More specifically, the model successfully identified 11,079 of the 13,140 helpful reviews (84.3%) and 5,371 of the 7,100 unhelpful reviews (75.6%) in the validation data.

To further assess the model and to demonstrate the value that document projections were able to add, you can compare the predictive results with and without using the document projections as a predictor. The misclassification results of three candidate models can be compared to determine which has the most predictive value:

- SVM Model using document projections, star rating, and review length
- SVM Model using only document projections (no star rating or review length)
- SVM Model using only star rating and review length

For the sake of space, the code to generate all three models and compare their misclassification rates is not included in this paper. Table 2 is a Jupyter Notebook output table created to compare the misclassification rates of the three models at a cutoff of p=0.50. This table shows that the model including all variables did the best job at classifying reviews as helpful.

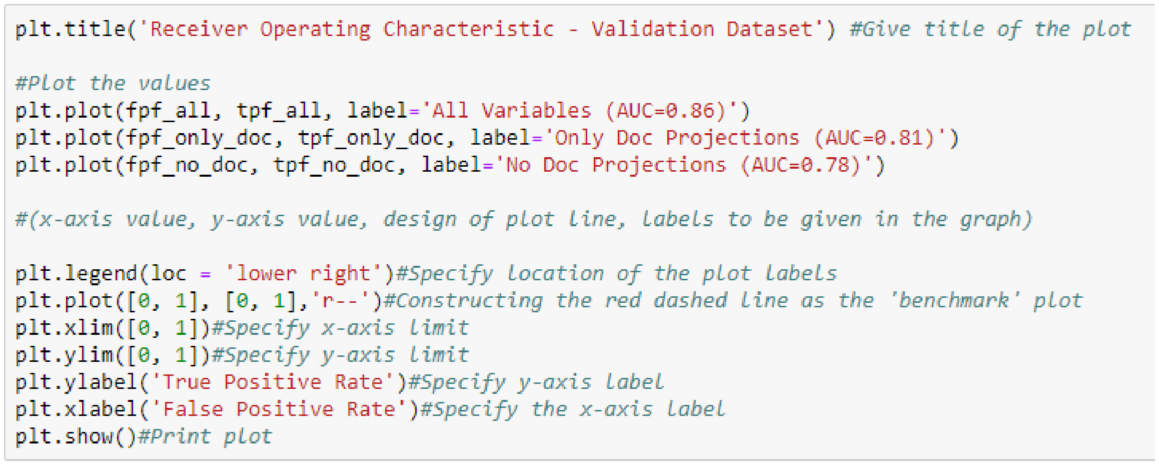


Misclassification Rate Comparison

Variable Scenarios	Misclassification Rates
With all Variables	0.187253
Without Doc Projections	0.223370
With Only Doc Projections	0.239427

Table 2. Comparing Misclassification Rates

For a visual comparison between models, the Matplotlib library can be used to generate an ROC curve for each model. Figure 20 shows how to generate the curves.



```
plt.title('Receiver Operating Characteristic - Validation Dataset') #Give title of the plot

#Plot the values
plt.plot(fpf_all, tpf_all, label='All Variables (AUC=0.86)')
plt.plot(fpf_only_doc, tpf_only_doc, label='Only Doc Projections (AUC=0.81)')
plt.plot(fpf_no_doc, tpf_no_doc, label='No Doc Projections (AUC=0.78)')

#(x-axis value, y-axis value, design of plot line, labels to be given in the graph)

plt.legend(loc = 'lower right')#Specify location of the plot labels
plt.plot([0, 1], [0, 1],'r--')#Constructing the red dashed line as the 'benchmark' plot
plt.xlim([0, 1])#Specify x-axis limit
plt.ylim([0, 1])#Specify y-axis limit
plt.ylabel('True Positive Rate')#Specify y-axis label
plt.xlabel('False Positive Rate')#Specify the x-axis label
plt.show()#Print plot
```

Figure 20. Code to Generate ROC Curves

The resulting ROC curves are shown in Figure 21.

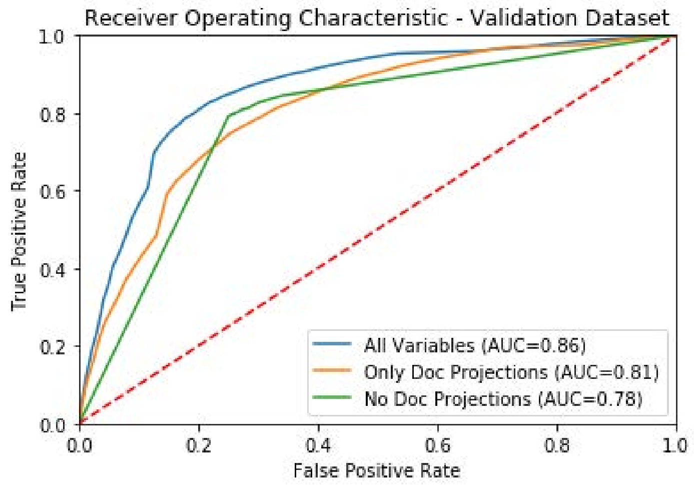


Figure 21. ROC Curves on Validation Data

The area under the curve improves significantly from 0.78 to 0.86 when document projections are included in the model. This demonstrates the value added through the text analytics model building process where unstructured text was converted into numeric input variables for use in the SVM model.

CONCLUSION

The SWAT library makes it easy for Python users to access and interact with the SAS Viya platform. Python users are able to write code in a familiar environment while gaining access to SAS text analytics and machine learning CAS actions. These actions were used throughout the course of this paper to highlight an approach for building an end-to-end text analytics model in SAS Viya using Python.

The value of incorporating unstructured text as input into a machine learning model was demonstrated. Amazon reviews were transformed into a numerical representation via the document projections output from the tmSvd action. Those document projections were combined with the star rating and review length as input into an SVM model to predict whether users will rate a review as being helpful. The results on the validation data proved the value of incorporating the document projections as they improved the misclassification rate and area under the curve significantly. The area under the curve from this model built with a traditional text mining approach plus an SVM can serve as a baseline for comparison against other techniques such as deep learning, an approach SAS makes accessible to Python users through the SAS Deep Learning Python (DLPy) package.

REFERENCES

Albright, R. 2004. SAS Institute white paper. “Taming Text with the SVD.” ftp.sas.com/techsup/download/EMiner/TamingTextwiththeSVD.pdf.

Foreman, Carrie. 2019. “SWAT’s it all about? SAS® Viya® for Python Users.” *Proceedings of the SAS Global Forum 2019 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3610-2019.pdf

Indelicato, Joe. 2019. "Open Visualization with SAS® Viya® and Python." *Proceedings of the SAS Global Forum 2019 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3455-2019.pdf

SAS Institute Inc. 2020. "What's New in SAS Visual Text Analytics 8.5" *SAS® Visual Text Analytics 8.5: Programming Guide*. Cary, NC: SAS Institute Inc. https://go.documentation.sas.com/?docsetId=casvtapg&docsetTarget=p0vouc3o8s7gq0n1p1b68jydb8id.htm&docsetVersion=8.5&locale=en

SAS Institute. 2020. "Support Vector Machine Action Set: svmTrain Action." *SAS® Visual Data Mining and Machine Learning 8.5: Programming Guide*. Cary, NC: SAS Institute Inc. https://documentation.sas.com/?docsetId=casactml&docsetVersion=8.5&docsetTarget=cas-svm-svmtrain.htm&locale=en

SAS Institute. 2020. "Percentile Action Set: assess Action." *SAS® Visual Analytics 8.5: Programming Guide*. Cary, NC: SAS Institute Inc. https://documentation.sas.com/?docsetId=casanpg&docsetTarget=cas-percentile-assess.htm&docsetVersion=8.5&locale=en

SAS Institute Inc. 2016. SWAT Documentation. https://developer.sas.com/apis/swat/python/v1.1.0/install.html

SAS Institute Inc. 2020. SAS Deep Learning Python (DLPy). https://github.com/sassoftware/python-dlpy

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Nate Gilmore
Nate.Gilmore@sas.com

Vinay Ashokkumar
Vinay.Ashokkumar@sas.com

Russell Albright
Russell.Albright@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies

Paper SAS2225-2018

Harvesting Unstructured Data to Reduce Anti-Money Laundering (AML) Compliance Risk

Austin Cook and Beth Herron, SAS Institute Inc.

ABSTRACT

As an anti-money laundering (AML) analyst, you face a never-ending job of staying one step ahead of nefarious actors (for example, terrorist organizations, drug cartels, and other money launderers). The financial services industry has called into question whether traditional methods of combating money laundering and terrorism financing are effective and sustainable. Heightened regulatory expectations, emphasis on 100% coverage, identification of emerging risks, and rising staffing costs are driving institutions to modernize their systems. One area gaining traction in the industry is to leverage the vast amounts of unstructured data to gain deeper insights. From suspicious activity reports (SARs) to case notes and wire messages, most financial institutions have yet to apply analytics to this data to uncover new patterns and trends that might not surface themselves in traditional structured data. This paper explores the potential use cases for text analytics in AML and provides examples of entity and fact extraction and document categorization of unstructured data using SAS® Visual Text Analytics.

INTRODUCTION

Financial Institutions dedicate substantial resources in support of government's efforts to curb money laundering and terrorism financing. Money laundering is the process of making funds that were gained through illegal channels appear legitimate, typically through a process of placement, layering, and integration. Terrorism financing is often more challenging to identify, as the funding can be raised through legitimate means, but later used to fund an act of terror or support a terrorist organization. Detecting these patterns can often feel like a game of "whack-a-mole;" by the time a new control is implemented to identify a known risk, the criminals have already changed their behavior to elude your efforts. The stakes are high, as the amount of money laundered per year is estimated to be 2 to 5% of global GDP. That's 2 trillion in USD according to the United Nations Office on Drugs and Crime (UNODC). In today's big-data environment, using modern technology to quickly identify financial crimes is critical.

A lot has changed globally since the early AML regimes of the 1970s. A growing regulatory landscape has led to higher penalties for program deficiencies. Banking has fundamentally changed with the creation of digital channels, faster payments, and new financial instruments. Data storage has become cheaper, opening the opportunity to process big data rapidly. Financial institutions have mostly adapted to these changes through enhancements to their rule-based detection programs and, as a result, have seen their headcount and costs soar. There's an appetite to overhaul the system to reduce false positive rates, increase the detection of money laundering, and automate many of the tedious tasks required in the investigations process. With the help of SAS® Visual Text Analytics, we can leverage artificial intelligence techniques to scale the human act of reading, organizing, and quantifying free-form text in meaningful ways, uncovering a rich source of underused risk data.

UNSTRUCTURED DATA SOURCES

While structured data such as transaction, account, and demographic information has been used in combating money laundering for years, financial institutions are just now beginning to see the value in harvesting unstructured data sources. These data sources are both vast and rich with valuable information that provides new data points, creates linkages, and identifies trends. Here is a list of the more notable sources of unstructured data that can be used for AML:

- **Wire Data -** Wire transfers between financial institutions contain much more valuable information than just the amount of money being sent. Along with origination, intermediary, and beneficiary data, wires often include free-form text including payment instructions and other messaging.

- **Transaction Review Memos** - The branch employees and client managers are the first line of defense when it comes to protecting the bank from money laundering. Typically, these individuals report valuable insight to the AML group through a transaction review memo. The details included in these memos are at the branch attendee's discretion, but often they have supporting detail on why the transaction was deemed suspicious that might not be apparent in the transaction alone.
- **Case Data** - Anti-money laundering case data contains information enriched by the investigator during the life of the investigation. Cases generally contain several free-form text fields including notes, comments, and email correspondence as well as a narrative report explaining the final disposition. If suspicious activity is identified, a suspicious activity report (SAR) will be filed.
- **Suspicious Activity Report Data** - SARs are documents that financial institutions must file with their in-country government agency following the identification of potential unusual behavior related to money laundering or fraud. These documents are typically free-form text and generally contain several pieces of information about the person, company, and entity or entities of interest; the general findings from the investigator as to what the suspicious activity was; as well as any supporting evidence for the suspicious activity.
- **Negative News** - Beyond unstructured data your financial institution generates, there is a vast amount of publicly generated data from news and media organizations. Public news data can be used to identify supporting information about your customers including relationships to businesses or risky behaviors and criminal activity.
- **Email/Phone/Chat** - In addition to transactional data, risk factors might be identified in the non-transactional data stored by the bank in the form of email, phone, or chat conversations between the customer and employee.
- **Law Enforcement Requests** - Financial institutions have an obligation to identify subjects of law enforcement requests and file SARs where appropriate. Grand jury subpoenas, national security letters, and other requests are received in electronic format and contain text regarding persons of interest and requests for information.
- **Trade Documents** - The global trade system remains primarily a paper-based system. The trade documents (letters of credit, bills of lading, commercial invoices, other shipping documents) contain critical risk information in free-form text such as boycott language, dual use goods, inconsistent unit pricing, and other trade-based, money-laundering vulnerabilities.

USE CASES IN AML

Mining your unstructured data can be valuable in uncovering new insights to help combat money laundering in your financial institutions. Processing techniques such as theme detection, categorization, and entity or fact extraction are all ways to provide structure to free-form text. Once text is structured, there are several use cases to apply this data to ensure compliance:

- **Negative News Monitoring** - As an industry standard, financial institutions typically look for negative news related to high-risk customers and customers who have an open AML case. With the wide array of digital news made available daily, the identification of credible news can be challenging. Negative news not relevant to compliance can bias an investigator's decision process, while missed news can leave an institution open to reputational risk. Coupled with bank policy and risk tolerance, an automated process to identify negative news and successfully link this information to customers provides both cost and time savings through automation.
- **Network Analytics** - Perhaps one of the best pieces of information for investigating AML is to understand relationships among your customers, as well as non-customers. Most institutions have structured data for known relationships among their customers, but often there are gaps with unknown relationships and those relationships with non-customers. Relationships and networks often surface through normal investigative procedures and are documented in case notes and SAR data. Storing this valuable information and displaying it for future use along with geographic tagging

provides deeper insights to the investigations process.

- **SAR Attribution Detection -** The detection of money laundering is an exercise in correctly identifying rare events in vast amounts of data. As the AML compliance industry starts to explore the application of artificial intelligence and machine learning to replace Boolean rules, the need for reliably labeled data (target variables) for training becomes even more important. Often, SARs are filed based on external information, but are attributed to the success of one or more rule-based scenarios. Text mining can help determine the correlation. This is critical to not only tune existing models, but also to allow banks to predict newly identified patterns in the future.
- **Trade Finance Document Categorization -** Deciphering trade documents is a tedious, manual process. We've been testing cognitive computing capabilities that are used for character recognition and natural language processing for document categorization. In a pilot with a tier 1 bank, our models read trade finance documents with ~99% accuracy and reduced the time to manually process the documents from several weeks to 26 seconds in an automated process.

EXAMPLE FRAMEWORK USING SAS® VISUAL TEXT ANALYTICS

This paper explores the process of processing unstructured data to support any of the use cases listed above. To demonstrate the potential applications, we will follow the framework below, primarily using SAS Visual Text Analytics as the enabling technology.

- **Data Acquisition –** Data is acquired for the example use case utilizing web scraping tools and is imported into SAS Visual Text Analytics.
- **Concept Extraction** – Predefined and customized concepts are generated to extract key facts from the unstructured data.
- **Text Parsing** – The individual records are parsed to enumerate the terms contained in the documents and apply filtering with start and stop lists.
- **Topic Generation** – Individual records are grouped into a collection of related themes containing similar subject matter automatically based on a bottom-up approach using the underlying terms.
- **Categorization** – Documents are classified into predetermined categories based on a top-down approach of the areas of interest using linguistic rules.
- **Post-Processing** – Output from SAS Visual Text Analytics is processed and prepared for use in modeling or investigative tools.

DATA ACQUISITION

While SAR information is not publicly available, we wanted to conduct our analysis on text data with similar content and format. The Internal Revenue Service (IRS) publishes summaries of significant money laundering cases each fiscal year, dating back to 2015. This data is rich with information, including people, organizations, risk typologies, locations, and other interesting data related to financial crimes. Below is an example of an IRS case from our data set:

"**Former Owners of Money Transmitter Business Sentenced for Conspiring to Structure Financial Transactions**
On October 25, 2016, in Scranton, Pennsylvania, German Ossa-Rocha was sentenced to 27 months in prison and two years of supervised release. On October 26, 2016, Mirela Desouza was sentenced to 18 months in prison and two years of supervised release. Ossa-Rocha and Desouza were the former owners of Tropical Express, a money transmitter service business located in Stroudsburg. Beginning in approximately January of 2008 and continuing through December 2011, Ossa-Rocha and Desouza structured financial transactions that represented the proceeds of drug trafficking in a manner intended to avoid reporting and recording requirements. The amount of funds involved in the structuring was approximately $340,000. The funds were transmitted by Ossa-Rocha and Desouza via wire transfers to the Dominican Republic." (IRS)

Web scraping tools were used to extract the various money laundering examples and write to a CSV file with four columns: observation number, year, title, and text narrative. The CSV file was then imported into SAS Visual Text Analytics for analysis.

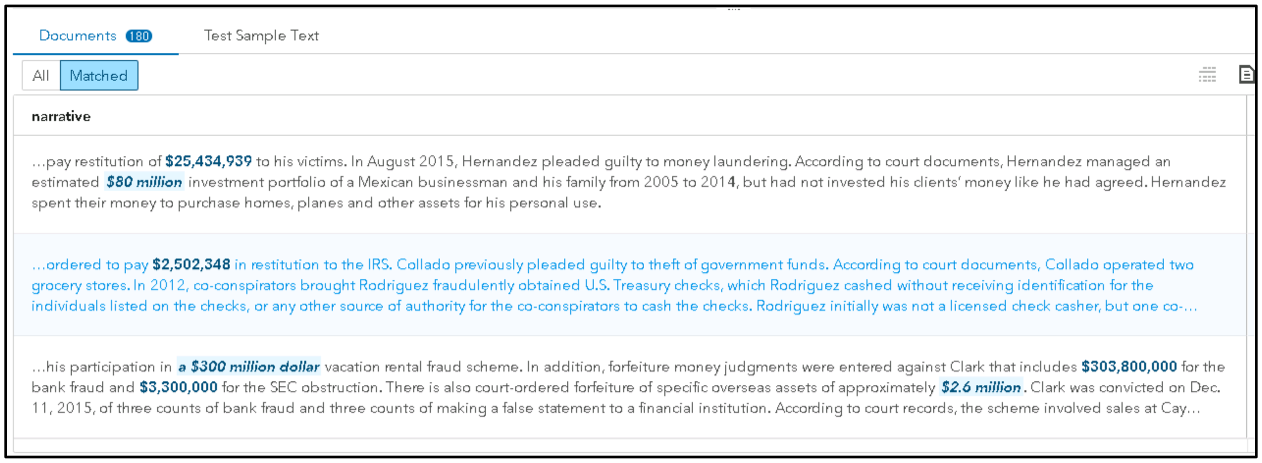
CONCEPT EXTRACTION

After initializing a project and loading the data, the first step in the process was focused on concept and fact extraction. With our data being rich in entities and facts, we wanted to extract these from the text for potential use in further analysis and research by investigators. In our model pipeline, this was done by dragging a Concept node and placing it on top of the Data node. SAS Visual Text Analytics comes with predefined concepts out of the box, as well as the ability to write your own custom concepts using LITI (language interpretation and text interpretation) syntax. For our analysis, we enabled the predefined concepts and wrote several custom concepts that are highlighted below.

The predefined concepts are common points of interest in which the rules come out of the box to immediately apply to your data, saving you time and helping you gain instant insights. Here are the predefined concepts of interest for our analysis:

- **nlpDate –** Identifies and extracts all dates and date ranges in your data in several formats (for example, May 2003, 05/15/2007, between 2007 and 2009, and so on).
- **nlpMeasure –** Identifies and extracts measures of time and quantities (for example, 30 years, 500 kilograms, and so on).
- **nlpMoney –** Identifies and extracts all references to currencies (for example, $272,000, more than $3 million, and so on).
- **nlpOrganizations –** Identifies and extracts all organization names (for example, U.S. Treasury, Department of Agriculture, and so on).
- **nlpPerson –** Identifies and extracts all names (for example, Joyce Allen, Robert L. Keys, and so on).
- **nlpPlace –** Identifies and extracts all places (for example, Asheville, North Carolina, Newport Beach, California, and so on).

Error! Reference source not found. below shows a set of matched concepts for the predefined concept nlpMoney.



narrative

…pay restitution of **$25,434,939** to his victims. In August 2015, Hernandez pleaded guilty to money laundering. According to court documents, Hernandez managed an estimated ***$80 million*** investment portfolio of a Mexican businessman and his family from 2005 to 2014, but had not invested his clients' money like he had agreed. Hernandez spent their money to purchase homes, planes and other assets for his personal use.

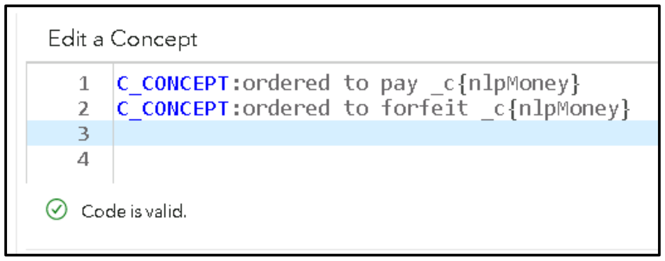
…ordered to pay **$2,502,348** in restitution to the IRS. Collado previously pleaded guilty to theft of government funds. According to court documents, Collado operated two grocery stores. In 2012, co-conspirators brought Rodriguez fraudulently obtained U.S. Treasury checks, which Rodriguez cashed without receiving identification for the individuals listed on the checks, or any other source of authority for the co-conspirators to cash the checks. Rodriguez initially was not a licensed check casher, but one co-…

…his participation in ***a $300 million dollar*** vacation rental fraud scheme. In addition, forfeiture money judgments were entered against Clark that includes **$303,800,000** for the bank fraud and **$3,300,000** for the SEC obstruction. There is also court-ordered forfeiture of specific overseas assets of approximately ***$2.6 million***. Clark was convicted on Dec. 11, 2015, of three counts of bank fraud and three counts of making a false statement to a financial institution. According to court records, the scheme involved sales at Cay…

Figure 1. Matched Concepts for Predefined Concept nlpMoney

While the predefined concepts are valuable in and of themselves, they are also useful for referencing in your custom concepts. An example of this can be seen with our custom concept Fine_Amount. The predefined concept nlpMoney will extract out all references to money, but suppose we want to exclusively extract out the fines associated with each record for further analysis. Instead of filtering through all references to money, we can define a custom concept to pull out only currencies associated with a fine.

Figure 2 below shows the LITI syntax to generate this rule:


Edit a Concept

```
C_CONCEPT:ordered to pay _c{nlpMoney}
C_CONCEPT:ordered to forfeit _c{nlpMoney}
```

Code is valid.

Figure 2. Custom Concept Fine_Amount LITI Syntax

The Fine_Amount custom concept uses the C_CONCEPT rule, which enables you to return matches that occur only in the context that we desire. In our case, we want to return the currency found by the nlpMoney predefined concept, but only in the context of a fine as in “ordered to pay” or “ordered to forfeit”.

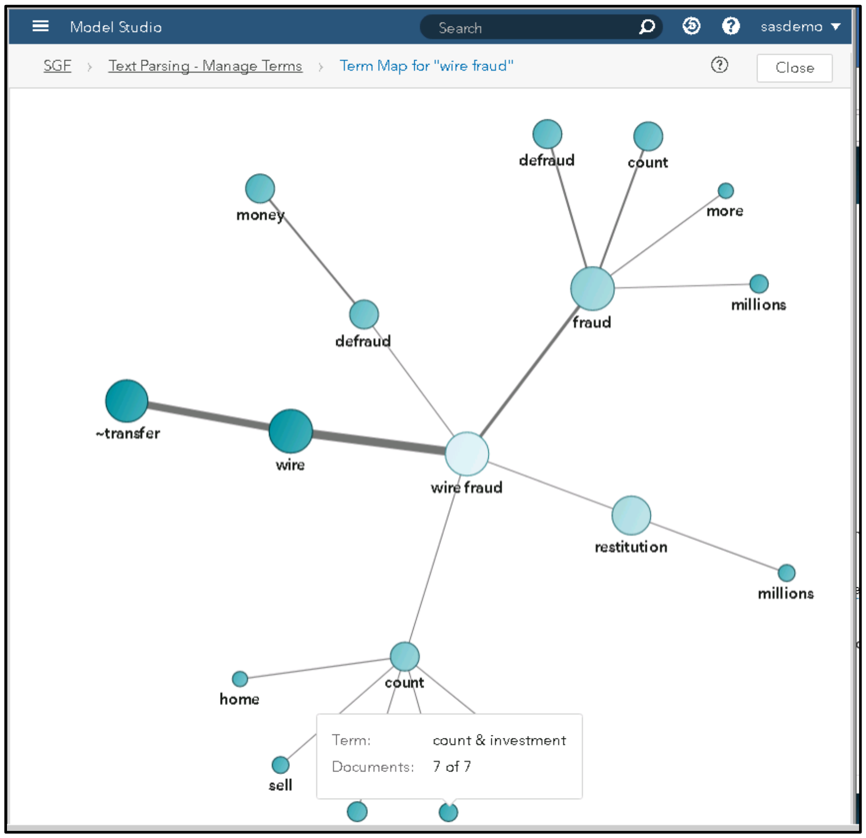
A set of custom concepts was built on top of the predefined concepts to extract additional useful facts that could be helpful for indexing and searching, as well as additional analysis. Table 1 below summarizes the custom concepts that were developed, the type of concept used, and an example of the output.

Custom Concept	Concept Type	Example Output
Drug_Names	CLASSIFIER	Marijuana
Prison_Sentence	C_CONCEPT	60 months
Drug_Amount	CONCEPT_RULE	15 kilograms
Investment_Fraud_Amount	CONCEPT_RULE	$200 million
Investment_Fraud_Victims	CONCEPT_RULE	70 victims
Case_Charges	CLASSIFIER	Identity theft
Sentence_Location	CONCEPT_RULE	Providence, Rhode Island

Table 1. Custom Concept Definitions

TEXT PARSING

The next step in our analysis was to parse the text and create our term document matrix. In our model studio pipeline, this is done by dragging the Text Parsing node and placing it on top of the Concept node. SAS Visual Text Analytics allows you to customize how terms are parsed by configuring the minimum number of documents the term must be found in to be included for analysis, as well as using custom start, stop, and synonym lists. For the purposes of our example, we used the Text Parsing node to further explore some terms of interest for additional context and understanding. Figure 3 is an example of a term map used for exploration purposes.

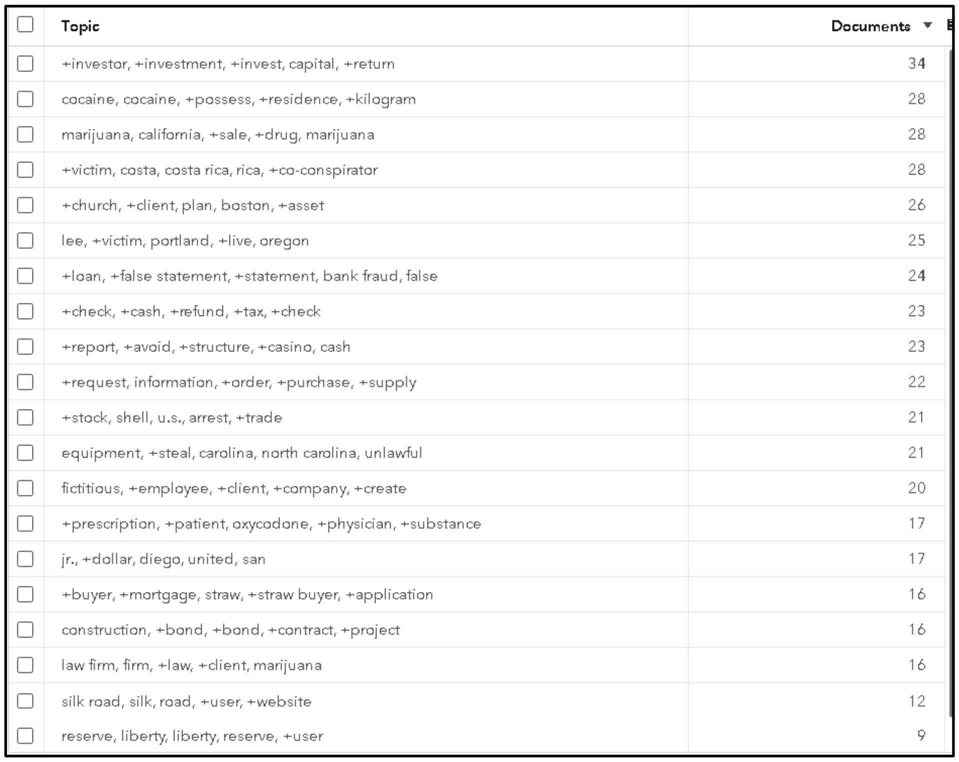


Term: count & investment
Documents: 7 of 7

Figure 3. Term Map for “wire fraud”

TEXT TOPICS

Continuing with our analysis, we wanted to understand any relevant themes found in the data with the underlying terms that were parsed. For this, we dragged a Topic node and placed it on top of the Text Parsing node. SAS Visual Text Analytics allows you to automatically generate topics or choose the number of topics to generate, as well as set several other configurations including the term and document density. With a few iterations, we found the most informative results by setting the number of topics generated at 20, as well as term and document density of 2 and 1, respectively. Here is the output of the text topics.



☐	Topic	Documents ▼
☐	+investor, +investment, +invest, capital, +return	34
☐	cocaine, cocaine, +possess, +residence, +kilogram	28
☐	marijuana, california, +sale, +drug, marijuana	28
☐	+victim, costa, costa rica, rica, +co-conspirator	28
☐	+church, +client, plan, boston, +asset	26
☐	lee, +victim, portland, +live, oregon	25
☐	+loan, +false statement, +statement, bank fraud, false	24
☐	+check, +cash, +refund, +tax, +check	23
☐	+report, +avoid, +structure, +casino, cash	23
☐	+request, information, +order, +purchase, +supply	22
☐	+stock, shell, u.s., arrest, +trade	21
☐	equipment, +steal, carolina, north carolina, unlawful	21
☐	fictitious, +employee, +client, +company, +create	20
☐	+prescription, +patient, oxycodone, +physician, +substance	17
☐	jr., +dollar, diego, united, san	17
☐	+buyer, +mortgage, straw, +straw buyer, +application	16
☐	construction, +bond, +bond, +contract, +project	16
☐	law firm, firm, +law, +client, marijuana	16
☐	silk road, silk, road, +user, +website	12
☐	reserve, liberty, liberty, reserve, +user	9

Figure 4. Text Topics and Associated Document Count

Upon inspecting the topics, we were interested in two themes that were promoted to categories for ongoing analysis. The topics that were automatically generated provided a new lens on the data that we would like to track further and categorize new documents moving forward.

Topic Terms	Topic Theme	Percent of Documents
+buyer, +mortgage, straw, +straw buyer, +application	Real Estate Investment Fraud	9.4%
silk road, silk, road, +user, +website	Dark Web Drug Trade	7.0%

Table 2. Text Topics Promoted to Categories

TEXT CATEGORIES

Previously, we discussed text topics and the bottom-up approach of using the underlying terms to generate topics of interest. Our next step in our analysis was to take a top-down approach and define categories of interest using linguistic rules available in SAS Visual Text Analytics. In our model pipeline, this is done by dragging a Category node and placing it on top of the Topic node.

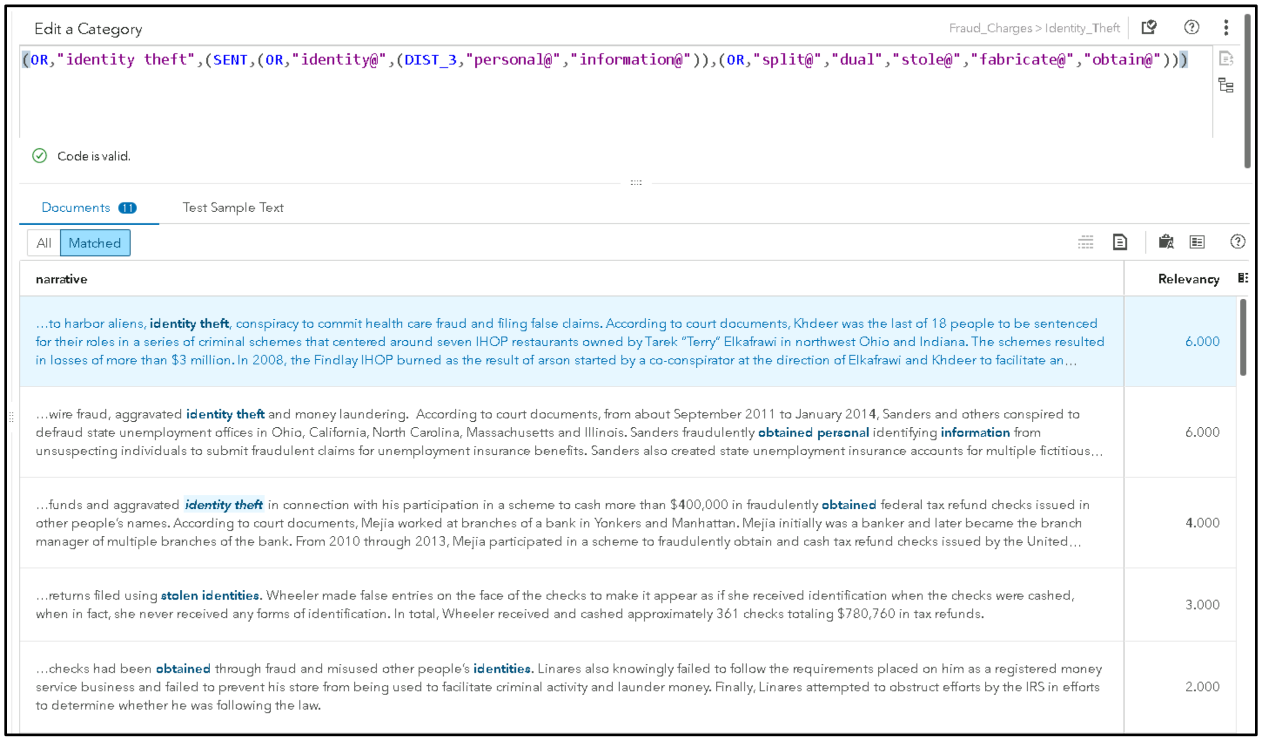
Categorizing your documents can be valuable for several reasons, such as creating tags for searching or for assigning similar documents for workflow purposes. Previously, we identified two categories of interest that we converted from the topics that were generated using the Topic node. In addition to these, we created a custom hierarchy of categorization that will help with future analysis. The table below shows the hierarchy of categories we were interested in.

Level 1	Level 2	Percentage of Matches
Drug Activity	Pharma Drugs	3%

	Illegal Drugs	15%
High Risk Customer Groups	Casino	3%
	Real Estate	23%
	Shell Company	3%
Financial Crime Charges	Bank Fraud	14%
	Bulk Cash Smuggling	4%
	Check Fraud	1%
	Identity Theft	6%
	Investment Fraud	8%
	Mail Fraud	16%
	Structuring	3%
	Tax Fraud	5%
	Wire Fraud	28%

Table 3. Custom Category Matches

Each category uses Boolean and proximity operators, arguments, and modifiers to effectively provide matches to only desired documents. Through the authors' domain expertise and the capabilities of SAS Visual Text Analytics, we were able to provide relevant matches on several categories of interest. An example of this concept is outlined below using the text category for the custom category "Identify Theft":



```
(OR,"identity theft",(SENT,(OR,"identity@",(DIST_3,"personal@","information@")),(OR,"split@","dual","stole@","fabricate@","obtain@")))
```

narrative	Relevancy
...to harbor aliens, **identity theft**, conspiracy to commit health care fraud and filing false claims. According to court documents, Khdeer was the last of 18 people to be sentenced for their roles in a series of criminal schemes that centered around seven IHOP restaurants owned by Tarek "Terry" Elkafrawi in northwest Ohio and Indiana. The schemes resulted in losses of more than $3 million. In 2008, the Findlay IHOP burned as the result of arson started by a co-conspirator at the direction of Elkafrawi and Khdeer to facilitate an...	6.000
...wire fraud, aggravated **identity theft** and money laundering. According to court documents, from about September 2011 to January 2014, Sanders and others conspired to defraud state unemployment offices in Ohio, California, North Carolina, Massachusetts and Illinois. Sanders fraudulently **obtained personal** identifying **information** from unsuspecting individuals to submit fraudulent claims for unemployment insurance benefits. Sanders also created state unemployment insurance accounts for multiple fictitious...	6.000
...funds and aggravated ***identity theft*** in connection with his participation in a scheme to cash more than $400,000 in fraudulently **obtained** federal tax refund checks issued in other people's names. According to court documents, Mejia worked at branches of a bank in Yonkers and Manhattan. Mejia initially was a banker and later became the branch manager of multiple branches of the bank. From 2010 through 2013, Mejia participated in a scheme to fraudulently obtain and cash tax refund checks issued by the United...	4.000
...returns filed using **stolen identities**. Wheeler made false entries on the face of the checks to make it appear as if she received identification when the checks were cashed, when in fact, she never received any forms of identification. In total, Wheeler received and cashed approximately 361 checks totaling $780,760 in tax refunds.	3.000
...checks had been **obtained** through fraud and misused other people's **identities**. Linares also knowingly failed to follow the requirements placed on him as a registered money service business and failed to prevent his store from being used to facilitate criminal activity and launder money. Finally, Linares attempted to obstruct efforts by the IRS in efforts to determine whether he was following the law.	2.000

Figure 5. Text Category for "Identity Theft" with Matched Output

The "Identity Theft" rule can be broken up into two main components using the OR operator. The first component is simply looking for a direct match for the two sequential terms "identity theft", which provides several simple matches in the output found in the bottom of Figure 5. The second component uses the

SENT operator and will trigger a match if two sub-components exist in the same sentence somewhere within the document. The first sub-component is looking for some form of the word “identity” or a close combination of “personal” and “information”. The second sub-component is looking for the action of theft including terms such as “split”, “dual”, “stole”, “fabricate”, or “obtain”. The fourth and fifth matches in Figure 5 highlight the types of matches this will create in the form of “stolen identities” and “obtained identities” in the fourth and fifth match, respectively.

POST-PROCESSING

Once your project is set up in SAS Visual Text Analytics, you can produce score code and apply this to new data for ongoing tracking and monitoring. There are several types of post-processing that can happen depending on your use case and what the type of output you are working with. The most common types of post-processing can be found below:

- **Categorical Flags** – Typically, the presence or match for a category is used as a binary indicator for each document and can be used in filtering or searching, or as inputs to machine learning algorithms.
- **Network Analysis** – Extracted concepts such as locations, people, and organizations can be post-processed to show linkages and used as input to network diagrams for analysis.
- **Numerical Analysis** – Extracted concepts such as duration, fine amounts, or other numerical fields extracted from the documents can be post-processed to derive summarizations and averages of areas of interest.

CONCLUSION

There is a lot of excitement in the financial crime and compliance industry around the application of artificial intelligence and automation techniques. We see many opportunities available today to apply these methods to improve the effectiveness of detection programs and automate the manual tasks being performed by investigators. Text analytics is one area that has enormous potential, given that compliance departments have vast amounts of untapped, unstructured data sources. These sources contain rich information including who, where, what, when, and how that can be used as an input to many financial crimes use cases such as Negative News Monitoring, Trade Finance Monitoring, and SAR/STR Quality Assurance. With SAS Visual Text Analytics, banks can extract and derive meaning from text and organize it in a way that helps them perform these complex tasks that were previously accessible only through manual human review.

REFERENCES

UNODC (United Nations Office on Drugs and Crime). *n.d.* “Money-Laundering and Globalization.” Accessed February 20, 2018. Available https://www.unodc.org/unodc/en/money-laundering/globalization.html.

IRS (Internal Revenue Service). 2017. “Examples of Money Laundering Investigations - Fiscal Year 2017.” Accessed February 20, 2018. Available https://www.irs.gov/compliance/criminal-investigation/examples-of-money-laundering-investigations-for-fiscal-year-2017.

ACKNOWLEDGMENTS

The authors would like to thank David Stewart for his guidance and thought leadership regarding AML compliance. In addition, we would like to thank Adam Pilz for his guidance and thought leadership regarding text analytics.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Austin Cook
100 SAS Campus Drive
Cary, NC 27513

SAS Institute Inc.
Austin.Cook@sas.com
http://www.sas.com

Beth Herron
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Beth.Herron@sas.com
http://www.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Paper SAS3213-2019

Hearing Every Voice: SAS® Text Analytics for Federal Regulations Public Commentary

Emily McRae, Tom Sabo, Manuel Figallo, SAS Institute Inc

ABSTRACT

Regulations.gov was launched in 2003 to provide the public with access to federal regulatory content and the ability to submit comments on federal regulations. Public participation in federal rulemaking is encouraged as it supports the legitimacy of regulatory decisions, frames public acceptance or resistance to rules under development, and shapes how the public interest will be served. Manually reading thousands of comments is time-consuming and labor-intensive. It is also difficult for multiple reviewers to accurately and consistently assess content, themes, stakeholder identity, and sentiment. Given that individually proposed rules can exceed 10,000 comments, how can federal organizations quantitatively assess the data and incorporate feedback into the rulemaking process as required by law?

This paper shows how SAS® Text Analytics can be used to develop transparent and accurate text models, and how SAS® Visual Analytics can quantify, summarize and present the results of that analysis. This will significantly decrease time to value, leveraging capabilities that computers excel at while freeing up human intuition for the analysis of these results. Specifically, we will address public commentary submitted in response to new product regulations by the US Food and Drug Administration. Ultimately, the application of a transparent and consistent text model to analyze these documents will support federal rule-makers and improve the health and lives of American citizens.

INTRODUCTION

Electronic cigarettes, or "e-cigarettes/e-cigs" are small electronic devices, which are used to simulate some of the properties of smoking. They work by heating a liquid to expel an aerosol or vapor, which the user then inhales. The heated liquid and resulting vapor in an e-cig typically contains nicotine, and the devices are very often marketed as a smoking cessation aid. Proponents of e-cig devices claim that they are healthier than traditional cigarettes, as they don't produce tar or noxious gases. However, it is important to note that a) nicotine is still a highly addictive substance, b) e-cigs might contain contaminants or other unintended toxins and c) the long-term health effects of e-cig usage are unclear. [1]

Of particular concern is the growing usage of e-cigarettes in children and other non-smokers. The National Youth Tobacco Survey (NYTS) is conducted annually by the FDA and CDC to track rates of tobacco use in children (middle and high school students). The most recent data indicates that usage is trending upward as the proportion of high-school users who reported using e-cigarettes on 20 days or more in the last 30-day period, increased from 20% to 27.7% between 2017-2018. Overall, approximately 3.6 million children reported using e-cigarettes in 2018. [2]

If children and teenagers are not using e-cigarettes to quit smoking, what is driving this trend? One suggestion is the inclusion of flavors in e-cigarette liquid, which replace the traditional tobacco taste of cigarettes. Common flavors include mint/menthol, strawberry, chocolate, cinnamon, gummy-bear, and cotton-candy, among many others.[3] For context, conventional cigarette manufacturers are prohibited from adding flavors (other than menthol) to their products, a move the FDA implemented specifically to reduce youth tobacco usage.[4]

To further understand the disparate drivers of e-cigarette usage and to balance their potential benefit as smoking cessation tools versus their potential risk to under-age users, the FDA submitted a notice of proposed rulemaking to *regulations.gov* (Regulation of Flavors in Tobacco Products, Docket ID: FDA-2017-N-6565) and opened a public comment period. The notice specifically asks for '*comments, data, research results, or other information about, … how flavors attract youth to initiate tobacco product use and about whether and how certain flavors may help adult cigarette smokers reduce cigarette use and switch to potentially less harmful products*.'[5]

This notice attracted over 525,000 comments, of which ~23,000 were made publicly available. Using this data, this paper will showcase a repeatable solution framework for the analysis of *regulations.gov* comments, in addition to familiarizing readers with the capabilities of SAS® Visual Text Analytics and SAS® Visual Analytics.

SOLUTION DEVELOPMENT

SAS® Visual Text Analytics provides a comprehensive suite of text mining capabilities, including Natural Language Processing (NLP), contextual extraction of concepts, categorization by machine learning and sentiment analysis. This paper will show how these capabilities can be used specifically for the analysis of *regulations.gov* data. Furthermore, aspects of this analysis can be used across different federal agencies and regulations, and so form the backbone of a repeatable solution.

SAS® Visual Text Analytics capability	***Regulations.gov* analysis**	**Repeatable?**
Concepts	Extract identity of referenced organizations	Y
Categories	Identify and quantify the document themes and form letters	N
Sentiment	Positive, Negative, or Neutral sentiment	Y

Table 1. SAS® Visual Text Analytics capabilities mapped to solution components

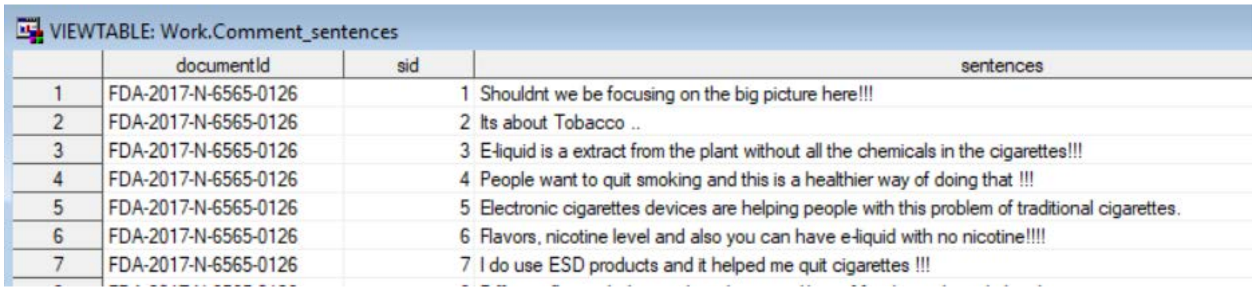
REGULATIONS.GOV API DATA EXTRACTION

Regulations.gov provides a public API for the extraction of documents (public comments) and dockets (the high-level organization folders, which contain the proposed regulations and any other background documents).[6] This paper uses code (courtesy of Manuel Figallo) to pull documents associated with Docket FDA-2017-N-6565.[7] The resulting data contains a unique ID for each comment, the comment text and other associated metadata, such as the posted date and title.

To prepare the data for analysis, we ran the following code:

```
/* Reformat Date variable and drop extraneous variables */
data work.reformat;
format Date Date9.;
set work._1_;
Date = input(postedDate, yymmdd10.);
drop agencyAcronym commentDueDate commentStartDate docketId docketTitle
postedDate;
run;
```

In addition, we ran sentence tokenizer code to break each individual comment into its constituent sentences and generate a sentence ID (sid).[8] This allows for more nuanced categorization of comments, as each comment could potentially reference multiple themes. The resulting tokenized data set is now ready for analysis in SAS® Visual Text Analytics.


VIEWTABLE: Work.Comment_sentences

	documentId	sid	sentences
1	FDA-2017-N-6565-0126	1	Shouldnt we be focusing on the big picture here!!!
2	FDA-2017-N-6565-0126	2	Its about Tobacco ..
3	FDA-2017-N-6565-0126	3	E-liquid is a extract from the plant without all the chemicals in the cigarettes!!!
4	FDA-2017-N-6565-0126	4	People want to quit smoking and this is a healthier way of doing that !!!
5	FDA-2017-N-6565-0126	5	Electronic cigarettes devices are helping people with this problem of traditional cigarettes.
6	FDA-2017-N-6565-0126	6	Flavors, nicotine level and also you can have e-liquid with no nicotine!!!!
7	FDA-2017-N-6565-0126	7	I do use ESD products and it helped me quit cigarettes !!!

Figure 1. Example *regulations.gov* analysis data set, following sentence tokenization

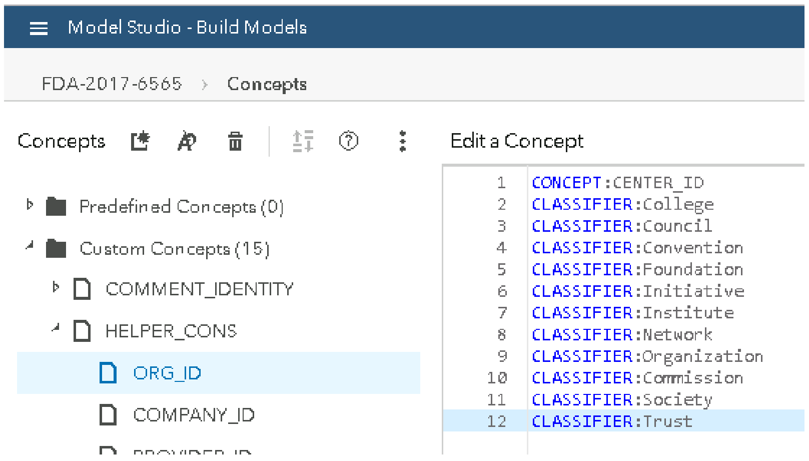
CONCEPTS FOR CONTEXTUAL EXTRACTION

Given that we know the subject of e-cigarette usage is fairly controversial, we decided to identify the different interest groups or identities associated with each comment. For example, is the comment poster an e-cigarette user? A doctor or other health care professional? Do they represent or quote from a special interest group or professional organization? How often are specific e-cigarette manufacturers mentioned?

These questions can be answered by using the concepts node in SAS® Visual Text Analytics, which uses contextual extraction rules to identify and extract specific entities (defined as people, places of things of interest, used here interchangeably with ‵concepts′). These contextual extraction rules use SAS proprietary LITI (Language Interpretation for Textual Information) syntax and use elements such as stemmed terms, parts-of-speech, and Boolean operators to define a specific context for a matched entity. This approach delivers high-precision results, especially in 'noisy' data.

IDENTIFY PROFESSIONAL ORGANIZATIONS

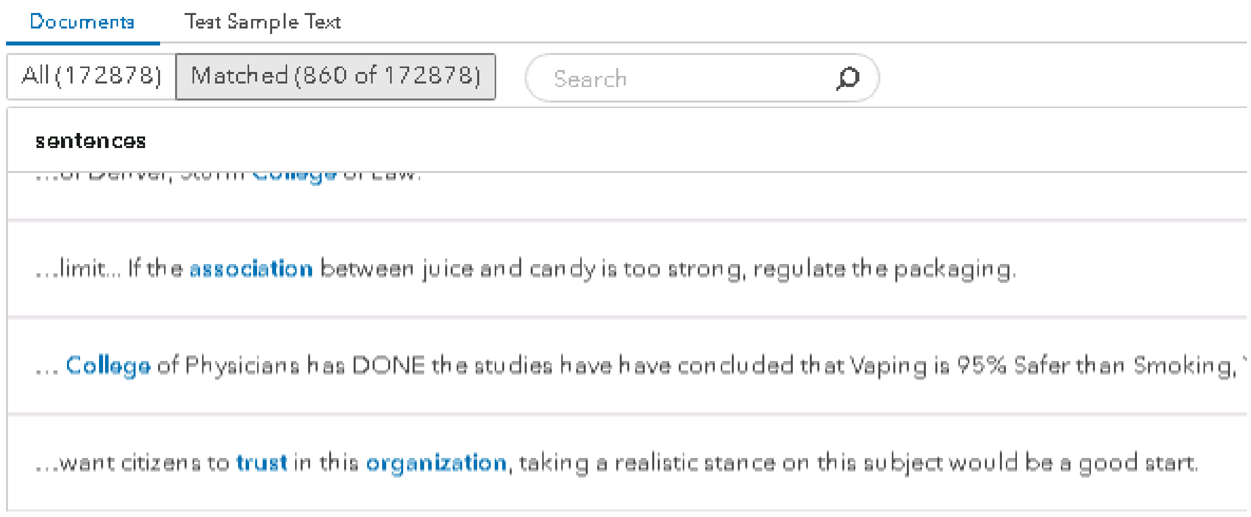
We initially noticed that many of these comments referred to research or positions stated by various professional organizations or lobby groups. When choosing to identify these organization entities, we first developed a series of simple rules, which define common organizational names. For example, the 'Academy of Such-and-Such' or the 'So-and-So Society', each one of these terms can be treated as a generic organizational 'flag.'



```
CONCEPT:CENTER_ID
CLASSIFIER:College
CLASSIFIER:Council
CLASSIFIER:Convention
CLASSIFIER:Foundation
CLASSIFIER:Initiative
CLASSIFIER:Institute
CLASSIFIER:Network
CLASSIFIER:Organization
CLASSIFIER:Commission
CLASSIFIER:Society
CLASSIFIER:Trust
```

Figure 2. Concept LITI rules define initial organization entity 'flags'

However, as you can see from the matched blue text in Figure 3 below, this approach returns noisy results, as many of these terms can have other meanings.



...of Denver, Sturm College of Law.

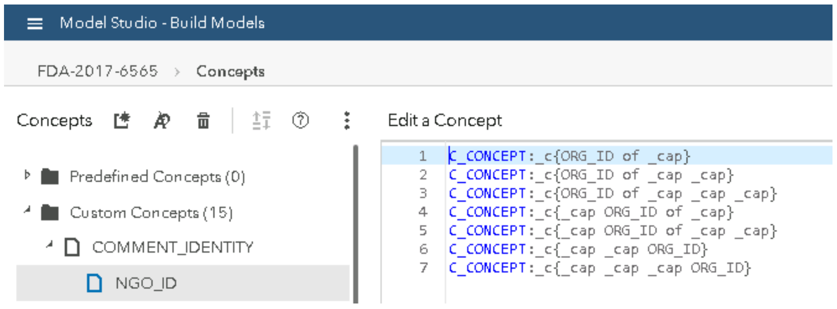
...limit... If the association between juice and candy is too strong, regulate the packaging.

... College of Physicians has DONE the studies have have concluded that Vaping is 95% Safer than Smoking,

...want citizens to trust in this organization, taking a realistic stance on this subject would be a good start.

Figure 3. Matched text for concept *'ORG_ID'* including irrelevant 'noise'

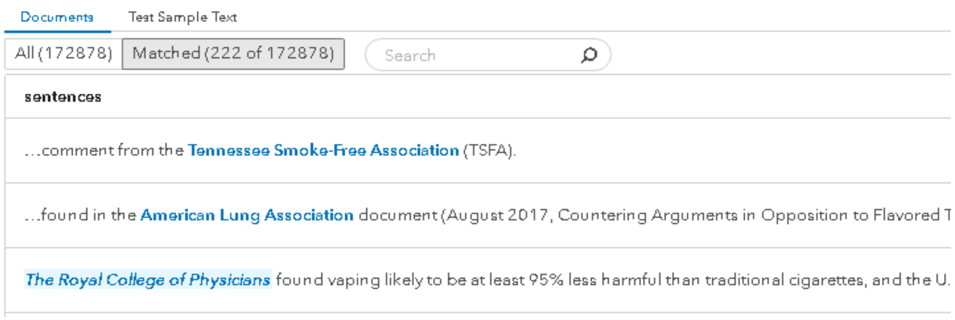
It is necessary to further restrict matches on these organizational terms to a particular context. In this case, we chose to build on these initial definitions and specified that they must reference capitalized terms (denoted by the _cap token) and occur in a particular structure, as below.



```
C_CONCEPT:_c{ORG_ID of _cap}
C_CONCEPT:_c{ORG_ID of _cap _cap}
C_CONCEPT:_c{ORG_ID of _cap _cap _cap}
C_CONCEPT:_c{_cap ORG_ID of _cap}
C_CONCEPT:_c{_cap ORG_ID of _cap _cap}
C_CONCEPT:_c{_cap _cap ORG_ID}
C_CONCEPT:_c{_cap _cap _cap ORG_ID}
```

Figure 4. Concept LITI definitions are refined to include structure and context

Now the matched blue text indicated that we are predominantly matching the organizational entities we are interested in.



...comment from the Tennessee Smoke-Free Association (TSFA).

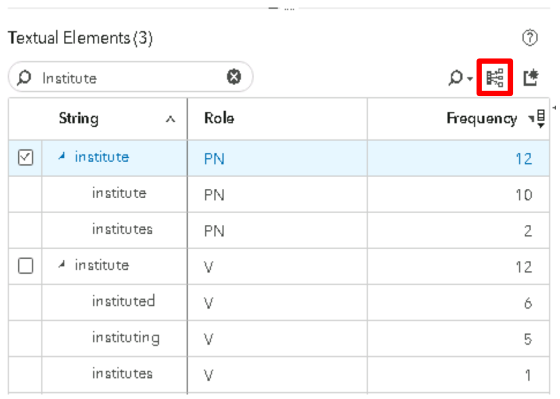
...found in the American Lung Association document (August 2017, Countering Arguments in Opposition to Flavored T

The Royal College of Physicians found vaping likely to be at least 95% less harmful than traditional cigarettes, and the U.

Figure 5. Matched text for concept *'NGO_ID'* showing improved accuracy

While we can generate accurate matches against organization entities, can we also improve the recall by adding in more definitions? And can we identify more organizational flags without reading through all the text?

SAS® Visual Text Analytics generates a 'similarity score', which is a measure of how likely it is for terms or phrases to occur in the same context. We used this feature to identify other organization flag terms. For example, using the 'Textual Elements' panel in the Concepts node, we searched for one of our flag terms – 'institute.'



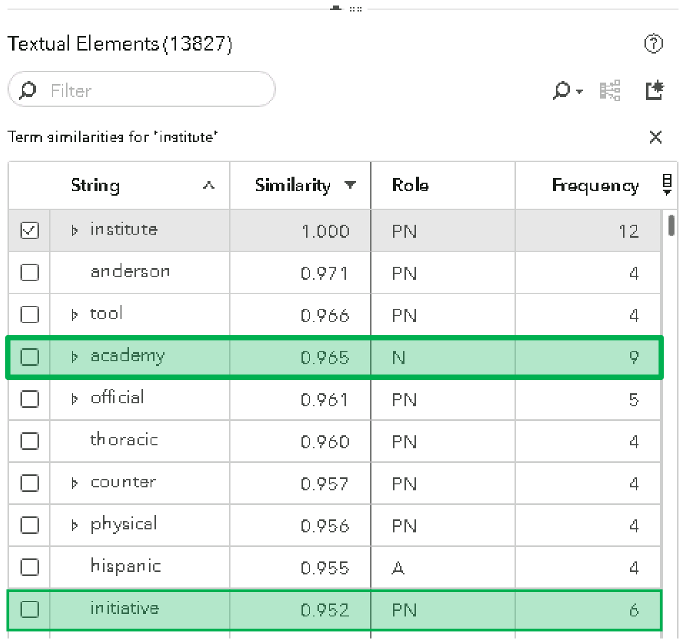
Textual Elements (3)

Institute

	String	Role	Frequency
✓	institute	PN	12
	institute	PN	10
	institutes	PN	2
	institute	V	12
	instituted	V	6
	instituting	V	5
	institutes	V	1

Figure 6. Frequency and Parts-of-Speech for filtered term 'Institute'

Note the 'Role' column indicated that this term is present as both a Proper Noun (PN) and a Verb (V). We selected the PN role and clicked on the 'Similarity Score' icon (highlighted in red box). This approach identifies two additional organization definitions ('academy' and 'initiative') that we can add to our original ORG_ID concept definitions, thereby increasing the number of positive matches.



Textual Elements (13827)

Filter

Term similarities for 'institute'

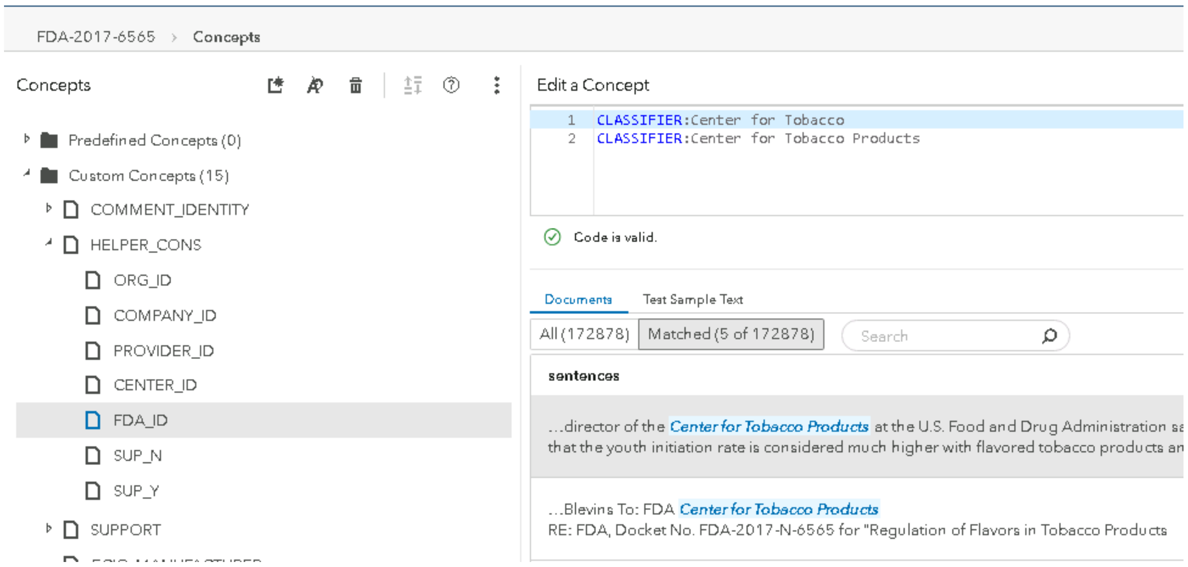
	String	Similarity	Role	Frequency
✓	institute	1.000	PN	12
	anderson	0.971	PN	4
	tool	0.966	PN	4
	academy	0.965	N	9
	official	0.961	PN	5
	thoracic	0.960	PN	4
	counter	0.957	PN	4
	physical	0.956	PN	4
	hispanic	0.955	A	4
	initiative	0.952	PN	6

Figure 7. Green highlighted fields show terms related to 'institute', following term similarity scoring

DISAMBIGUATE MATCHES

The preceding sections indicated how it is possible to identify and restrict matches for organization entities to a particular context. But what if we need additional control over these matches? For example, the flag word 'center' is one of the definitions used to identify organizations, but also matches references to the Center for Tobacco Products (CTP), which is the FDA Center responsible for producing this *regulations.gov* docket. Can we remove the organizational matches for CTP?

SAS® Visual Text Analytics provides a special LITI rule type called REMOVE_ITEM, which allows users to disambiguate matches with a common, partial definition, which can mean more than one thing. We start by explicitly specifying the matches we want to remove in the 'FDA_ID' concept:



```
CLASSIFIER:Center for Tobacco
CLASSIFIER:Center for Tobacco Products
```

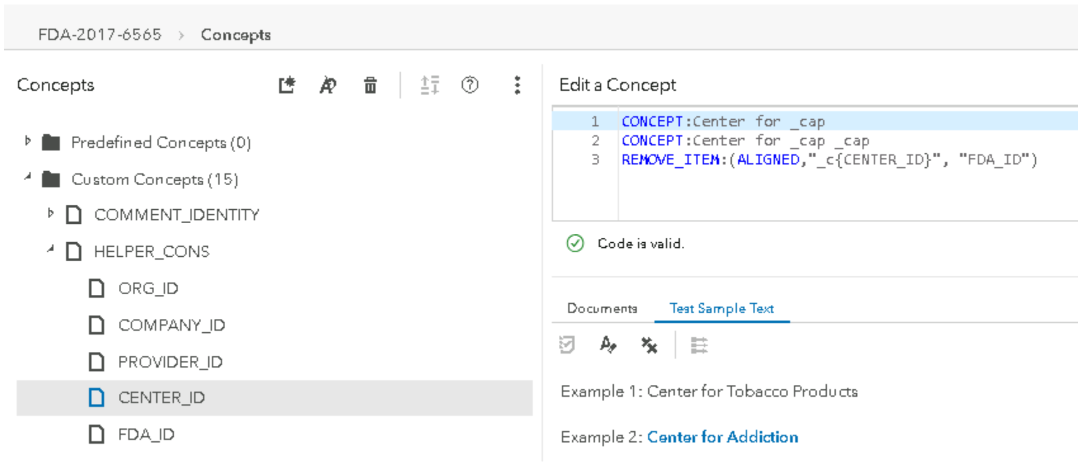
Code is valid.

...director of the Center for Tobacco Products at the U.S. Food and Drug Administration sa
that the youth initiation rate is considered much higher with flavored tobacco products an

...Blevins To: FDA Center for Tobacco Products
RE: FDA, Docket No. FDA-2017-N-6565 for "Regulation of Flavors in Tobacco Products

Figure 8. Concept LITI Classifier rules define unwanted matches

Then we use the 'ALIGNED' operator in a REMOVE_ITEM rule to return unrestricted matches on 'CENTER_ID' concepts, except where they also return 'FDA_ID' concept matches. We tested the results with some sample text, as seen below.



```
CONCEPT:Center for _cap
CONCEPT:Center for _cap _cap
REMOVE_ITEM:(ALIGNED,"_c{CENTER_ID}", "FDA_ID")
```

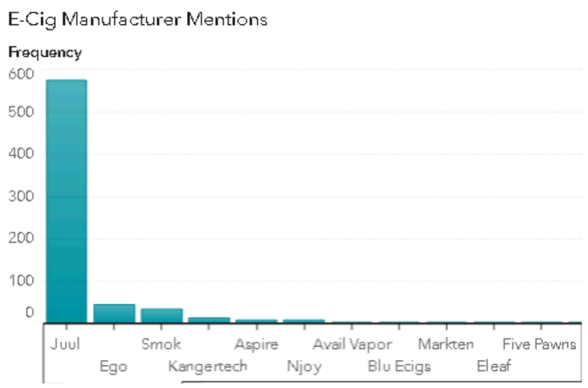
Code is valid.

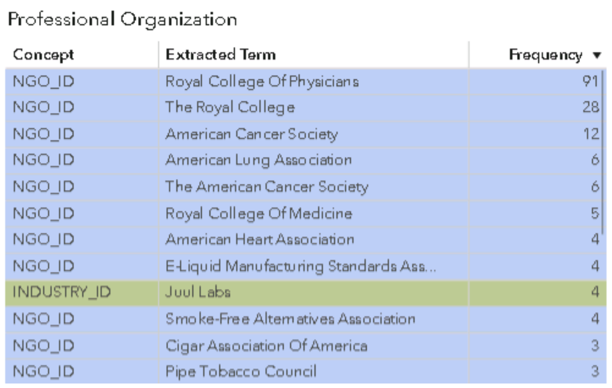
Example 1: Center for Tobacco Products

Example 2: Center for Addiction

Figure 9. The LITI REMOVE_ITEM rule provides disambiguation by removing unwanted matches

Following the development of our concept rules, we used the Concepts score code produced by SAS® Visual Text Analytics to extract mentions of e-cigarette manufacturers and professional organizations. This newly structured data was then used as input for report-building in SAS® Visual Analytics. The combination of SAS® Visual Text Analytics and SAS® Visual Analytics allows users to quickly visualize and interact with the results of text analysis. In particular, this approach makes the results of text modeling by a select few analysts available to a much broader audience of stakeholders.




Professional Organization

Concept	Extracted Term	Frequency ▼
NGO_ID	Royal College Of Physicians	91
NGO_ID	The Royal College	28
NGO_ID	American Cancer Society	12
NGO_ID	American Lung Association	6
NGO_ID	The American Cancer Society	6
NGO_ID	Royal College Of Medicine	5
NGO_ID	American Heart Association	4
NGO_ID	E-Liquid Manufacturing Standards Ass...	4
INDUSTRY_ID	Juul Labs	4
NGO_ID	Smoke-Free Alternatives Association	4
NGO_ID	Cigar Association Of America	3
NGO_ID	Pipe Tobacco Council	3

Figure 10. Juul is by far the most referenced e-cigarette manufacturer. The Royal College of Physicians and the American Cancer Society top the list of referenced professional organizations.

CATEGORIZATION FOR THEME IDENTIFICATION

One of the main challenges associated with text analysis is simply that of data volume. The point at which it becomes very intensive to manually read through text is reached after a comparatively small number of documents. Think of a spreadsheet with a few hundred observations of structured data (dates, prices, quantities, and so on), which can be quickly interpreted with a representative graphic, versus the same number of free-form text documents, each of which might refer to any topic in any form! In this instance, with over 20,000 comments publicly available, how can we quickly and comprehensively assess the representative themes?

SAS® Visual Text Analytics addresses this issue by using a 2-step machine learning approach to categorize documents by the main ideas or themes across any volume of text data. Initially, the Topics node generates a series of lists, each of which contain related terms. These terms commonly occur together in a subset of the model document corpus. Therefore, it can be used to describe these documents, as well as differentiate them from the rest of the documents in the model corpus. In the second step, the Categories node combines the selected terms from the Topics model with Boolean operators to generate linguistic rules, which can be used to assess document inclusion for a category/theme.

SMOKING CESSATION

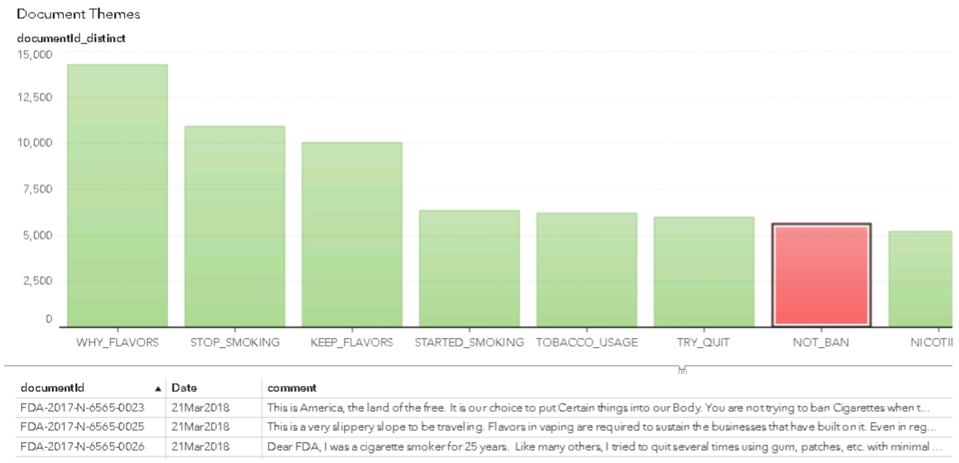
SAS® Visual Text Analytics categories automatically generated by the above machine-learning approach are initially labeled with a list of relevant terms. However, after reviewing the documents in each category, we chose to rename them in a more interpretable manner. It was very quickly apparent that the majority of opinions expressed in this data relate to the ways in which e-cigarettes, or specifically e-cig flavors impact people's desire or ability to quit smoking.

Category	Example
TOBACCO_USAGE	I was a 2 **pack** a **day smoker** for over 10 **years** and the **day** I picked up a vape is the **day** I was able to quit cigarettes.
WHY_FLAVORS	Most of the people I know quit smoking (who quit smoking rather) stopped by **using flavors** OTHER than **Menthol** and Tobacco **flavors**;
RESUME_SMOKING	But lets just say if you took are flavors away from us I would more than likely **go back** to **smoking**
TRY_QUIT	I have **tried several times** to **quit** smoking **tried** every method none of them **work** until I started vaping
STOP_SMOKING	many people have benefited from these flavors and have **quit smoking**
SAVE_LIVES	**Vaping** can **save lives** if I can stop smoking cigarettes after 20 years of a pack and a half a day and now I'm down to 3 mg of nicotine
CHILD_USAGE	would you rather have **kids vaping** or smoking a cig I would rather hear a **kid vaping** than smoking to be honest.

Table 2. Categorization of *regulatons.gov* comments by themes and example matching text

Interestingly, though most opinions seemed to suggest general endorsement of the role e-cigs play in smoking cessation, the categorical analysis indicates that commenters chose to frame their arguments in several ways, ranging from documenting their previous smoking habits, as well as their previous attempts to quit smoking, to suggesting a return to traditional cigarettes if flavors were banned. Most of the commenters appear to be adult e-cigarette users and any reference to the abuse of e-cigarette flavors by under-18s appears to be largely ancillary.

We used the Category and Sentiment score code produced by SAS® Visual Text Analytics to assign these documents to categories and to generate sentiment scores. This data was then used as input to a bar-chart in SAS® Visual Analytics. By clicking on the bars in the graphic below, it is also possible to toggle through the associated documents for each category. The addition of SAS® Visual Analytics to this approach greatly improves the interpretability of this data for the purposes of sharing with a wider audience.



documentId ▲	Date	comment
FDA-2017-N-6565-0023	21Mar2018	This is America, the land of the free. It is our choice to put Certain things into our Body. You are not trying to ban Cigarettes when t...
FDA-2017-N-6565-0025	21Mar2018	This is a very slippery slope to be traveling. Flavors in vaping are required to sustain the businesses that have built on it. Even in reg...
FDA-2017-N-6565-0026	21Mar2018	Dear FDA, I was a cigarette smoker for 25 years. Like many others, I tried to quit several times using gum, patches, etc. with minimal ...

Figure 11. The most prevalent themes in these documents express support for e-cigarette flavorings or discuss their usage as smoking cessation aids.

FORM LETTER IDENTIFICATION

In addition to the volume and variety of documents submitted to *regulations.gov*, it is also apparent that some individuals participate in concerted campaigns centered around a common point of view. In many cases, these individuals chose to submit form letters, in which most of the document is identical and can contain only small customizations, if any. It would be very difficult, if not impossible, to manually identify and quantify these form letters. However, by using the output from the SAS® Visual Text Analytics Topics node, which produces topic scores for each document, followed by a clustering analysis, it is possible to group together documents, which can be very similar or identical.

We ran the Text Topics score code, followed by some additional post-processing code to produce clusters:

```
/* PROC Distance generates measures of distance/similarity that can then be
used as input for clustering  */
proc distance data=casuser._out_documents out=Dist method=Euclid;
var interval(_Col1_-- _Col17_ / std=Std);
ID documentId;
run;

/* PROC Cluster groups related documents */
proc cluster data=Dist method=Ward outtree=tree;
ID documentId;
run;

/* PROC Tree generates cluster membership */
proc tree data=tree out=clusters nclusters=1400;
ID documentId;
run;

/* Remove clusters of only 1 document */
data work.clusters2;
set work.clusters;
where documentId ^= clusname;
run;

/* Join documentId and clusters to original text */
proc sql;
create table clusters3 as
select C.*, S.sentences
from work.clusters2 C left join public.fda_forms_score S
on C.documentId = S.documentId;
quit;
run;
```

This process generates a new data set, in which form letters are grouped together under the same cluster. The following table shows an example form letter, in which the general outline is the same, but individual points are customized by each poster. This approach allows us to identify groups of individuals who are commenting as a bloc, while also quantifying their contribution to *regulations.gov*.

Cluster CL1467	
FDA-2017-N-6565-18305	**FDA-2017-N-6565-18360**
Declaration of [@advFirst][@advLast]	Declaration of [@advFirst][@advLast]
I, [@advFirst][@advLast], declare and state as follows:	I, [@advFirst][@advLast], declare and state as follows:
1. I am over the age of 18 years and have personal knowledge of the facts set forth below such that I would be competent to testify as a witness to the same if called.	1. I am over the age of 18 years and have personal knowledge of the facts set forth below such that I would be competent to testify as a witness to the same if called.
2. I am submitting this declaration in response to the FDAs request for public comments on the above-referenced docket regarding a proposed rule on the regulation of flavors in tobacco products, including electronic nicotine delivery system (ENDS), or vapor products.	2. I am submitting this declaration in response to the FDAs request for public comments on the above-referenced docket regarding a proposed rule on the regulation of flavors in tobacco products, including electronic nicotine delivery system (ENDS), or vapor products.
3. I am a resident of [@advCityState] and am 48 years of age.	3. I am a resident of [@advCityState] and am 53 years of age.
4. I started smoking cigarettes when I was 18 years old and smoked them for more than 25 years .	4. I started smoking cigarettes when I was 14 years old and smoked them for more than 35 years .
5. I have been using vapor products for 5 years and 11 months.	5. I have been using vapor products for 3 years and 5 months.
6. The categories of flavors of nicotine-containing e-liquid that I have used include Menthol/Mint, Fruit, Desserts. Of these, the flavor category that I use most often is Fruit.	6. The categories of flavors of nicotine-containing e-liquid that I have used include Tobacco, Menthol/Mint, Fruit, Desserts, Other Sweets, Other Flavors. Of these, the flavor category that I use most often is Desserts.
7. Before I started using flavored e-liquid products, I typically smoked one pack of cigarettes per day.	7. Before I started using flavored e-liquid products, I typically smoked one pack of cigarettes per day.
8. E-cigarettes have helped my smoking cessation more than any other product or medication on the market to date. Having tried nicotine patches, gum, as well as	8. Over my thirty plus years of smoking (Marlboro Red, Camel Light, Marlboro Medium/27, Dunhill, and Sobrane)
[.... abbreviated for length...]	[.... abbreviated for length...]
9. Since I began using flavored e-liquid products, I have been able to quit smoking cigarettes....	9. Since I began using flavored e-liquid products, I have been able to quit smoking cigarettes....

Table 3. Example form letters. Text highlighted in green is identical. Remaining text is customized by individual poster.

CONCLUSION

The objective of this paper was to map SAS® Visual Text Analytics capabilities to the analysis of *regulations.gov* data, such that this approach can be replicated by readers for their own federal agencies or interests. Specifically, we showed how LITI contextual extraction can be used in SAS® Visual Text Analytics Concept rules to identify referenced organizations, manufacturers or comment poster identities. In addition, we showed how SAS® Visual Text Analytics Topic and Category modeling can be used to identify document themes and quantify form letters. This process is summarized in Figure 12, below.

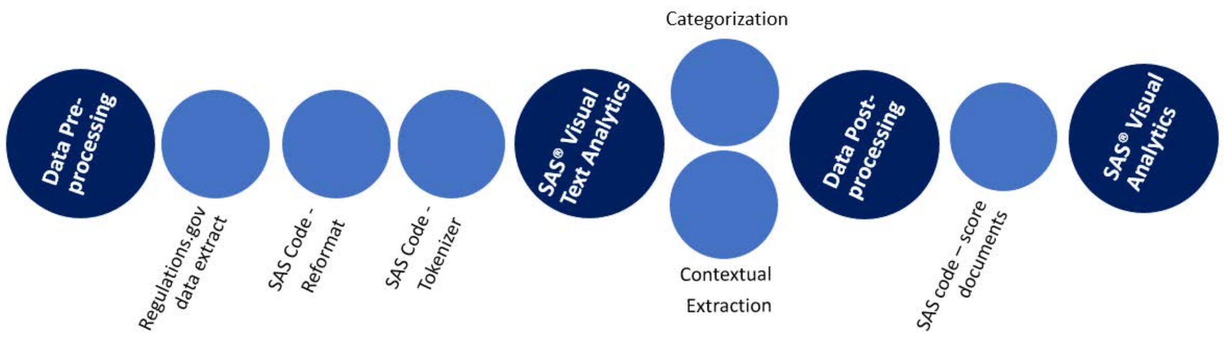


Figure 12. Overview of regulations.gov solution process

Regulations.gov is a very valuable source of text data and a key bridge between members of the public and the federal agencies, which serve their interests. Federal agencies are required by law to include the public in their rule-making process. According to section 553 of the Administrative Procedure Act, "*the agency shall give interested persons an opportunity to participate in the rule making through submission of written data, views, or arguments with or without opportunity for oral presentation.*" Traditionally, this has been accomplished by a manual review of public comment data, and in the case of controversial regulations, can involve reading thousands of documents to identify themes, sentiment and form letters.

SAS® Visual Text Analytics offers three key benefits for the analysis of *regulations.gov* data:

1. Consistency: human reviewers often struggle with ambiguity or nuanced definitions when assigning document themes. An algorithmic approach avoids these inconsistencies by generating a statistical representation of categories from term and document frequencies
2. Transparency: SAS® Visual Text Analytics models are not black-box, and results can be easily queried and interpreted. This supports the legitimacy of federal analysis for these comments and the overall rule-making process
3. Scalability: SAS® Visual Text Analytics models can be scaled to accommodate large volumes of data and category definitions. This is especially valuable as manual review of text is time-consuming and labor-intensive

This methodology can be applied to any public comments, as well as a wide variety of other text sources. The development and use of a text solution for *regulations.gov* analysis has the potential to greatly improve the accuracy and efficiency of these review efforts, particularly by reducing time to value. For example, if a docket receives 10,000 comments, and each comment takes 5 mins to manually review:

10,000 comments x 5 minutes = 833 hours / 20 FTE weeks

In contrast, the process documented in this paper took approximately a week to develop. This represents a significant time-saving improvement, with the opportunity for continued improvement as the reuseable solution is applied to more text.

REFERENCES

[1] E-cigarettes: Good news, bad news Available at: https://www.health.harvard.edu/blog/electronic-cigarettes-good-news-bad-news-2016072510010

[2] 2018 National Youth Tobacco Survey Available at: https://www.fda.gov/TobaccoProducts/PublicHealthEducation/ProtectingKidsfromTobacco/ucm625887.htm

[3] Flavored e-cigarette use: Characterizing youth, young adult, and adult users Available at: https://www.sciencedirect.com/science/article/pii/S2211335516301346

[4] Tobacco Control Act, section 907(a)(1)(A) Available at:

https://www.fda.gov/tobaccoproducts/guidancecomplianceregulatoryinformation/ucm263053.htm

[5] Docket FDA-2017-N-6565 Available at: https://www.regulations.gov/document?D=FDA-2017-N-6565-0001

[6] Regulations.gov Developers Info Available at: https://regulationsgov.github.io/developers/

[7] Regulations.gov API extraction code Available at: https://github.com/sasgovernment

[8] Sabo, T., Pilz, A. 2018 "Using SAS® Text Analytics to Assess International Human Trafficking Patterns"

ACKNOWLEDGMENTS

I would like to sincerely thank Tom Sabo and Manuel Figallo for their technical expertise and encouragement during my first SGF. Evelyn Fox and Tom Sabo reviewed my *regulations.gov* solution package and offered valuable feedback. I would also like to thank my team, particularly Lauren Laufe and Matt McTiernan, for their support.

RECOMMENDED READING

- *SAS® Visual Text Analytics 8.3: User's Guide*
- *SAS Text Analytics for Business Applications: Concept Rules for Information Extraction Models:* Jade, Teresa. Wilsey, Biljana Belamaric. Wallis, Michael. *Forthcoming 2019*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Emily McRae
SAS Institute
Emily.mcrae@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SAS 4180-2020

Multilingual Sentiment Analysis: An RNN-Based Framework for Limited Data

Ethem Can and Aysu Ezen-Can, SAS Institute Inc.

ABSTRACT

Sentiment analysis is a widely studied natural language processing task, whose goal is to determine users' opinions, emotions, and evaluations of a product, entity, or service that they review. One of the biggest challenges for sentiment analysis is that it is highly language-dependent. Word embeddings, sentiment lexicons, and even annotated data are language-specific. Furthermore, optimizing models for each language is very time-consuming and labor-intensive, especially for recurrent neural network (RNN) models. From a resource perspective, it is very challenging to collect data for different languages.

In this paper, we look for an answer to the following research question: Can a sentiment analysis model that is trained on one language be reused for sentiment analysis in other languages where the data are more limited? Our goal is to build a single model in the language that has the largest data set available for the task and reuse that model for languages that have limited resources.

For this purpose, we use reviews in English to train a sentiment analysis model by using recurrent neural networks. We then translate those reviews into other languages and reuse the model to evaluate the sentiments. Experimental results show that our robust approach of training a single model on English-language reviews outperforms the baseline in several different languages.

INTRODUCTION

Steady growth in commercial websites and social media venues has led to easier access to users' reviews. As the amount of data that can be mined for opinion has increased, commercial companies' interests in sentiment analysis has also increased. Sentiment analysis is an important part of understanding user behavior and opinions about products, places, or services.

Sentiment analysis has long been studied by the research community, leading to several sentiment-related resources such as sentiment dictionaries that can be used as features for machine learning models (Banea, Mihalcea, and Wiebe 2008; Inui and Yamamoto 2011; Steinberger et al. 2012; Taboada et al. 2011). These resources help increase the accuracy of sentiment analysis, but they are highly dependent on language and they require researchers to build such resources for every language to process.

Feature engineering constitutes a large part of the model-building phase for most sentiment analysis and emotion detection models (Ortigosa, Martin, and Carro 2014). Determining the correct set of features is a task that requires thorough investigation. Furthermore, these features are highly dependent on language and the particular data set, making it even more challenging to build models for different languages. For example, sentiment and emotion lexicons, as well as pretrained word embeddings, are not completely transferable to other languages. Therefore, modeling efforts must be replicated for every language on which you want to build sentiment classification models. For languages and tasks where the data are limited, extracting these features, building language models, training word embeddings, and creating lexicons are big challenges. In addition to the feature engineering effort, the machine learning model's parameters also need to be tuned separately for each language in order to obtain optimal results.

In this paper, we take a different approach. We build a reusable sentiment analysis model that does not use any lexicons. Our goal is to evaluate how well a generic model can be used to mine opinions in other languages where data are more limited than the language on which the generic model is trained. To that end, we train a recurrent neural network (RNN) model to predict polarity of reviews. To evaluate the reusability of the sentiment analysis model, we test with non-English data sets. We first translate the test set to English and use the pretrained model to score polarity in the translated text. In this way, our proposed approach eliminates the need to train language-dependent models for their use of sentiment lexicons and word embeddings. Our experiments show that a generalizable sentiment analysis model can be used successfully to perform opinion mining for languages that do not have enough resources for training specific models.

This study makes the following contributions:

- a robust approach that uses machine translation to reuse a model trained on one language in other languages
- an RNN-based approach to eliminate feature extraction and resource requirements for sentiment analysis
- a technique that outperforms baselines for multilingual sentiment analysis task when data are limited

METHODOLOGY

In order to eliminate the need to find data and build separate models for each language, we propose a multilingual approach in which a single model is built in the language that has the highest number of resources are available. In this paper, we focus on English for building a model because several sentiment analysis data sets are available in English. To make the English sentiment analysis model as generalizable as possible, we train with a data set that has reviews from different domains, such as camera reviews and restaurant reviews. To employ the trained model, test sets are first translated to English via machine translation and then inference takes place. Figure 1 shows our multilingual sentiment analysis approach.

It is important to note that this approach does not use any resource (such as word embeddings, lexicons, or a training set) in any of the languages of the test sets.

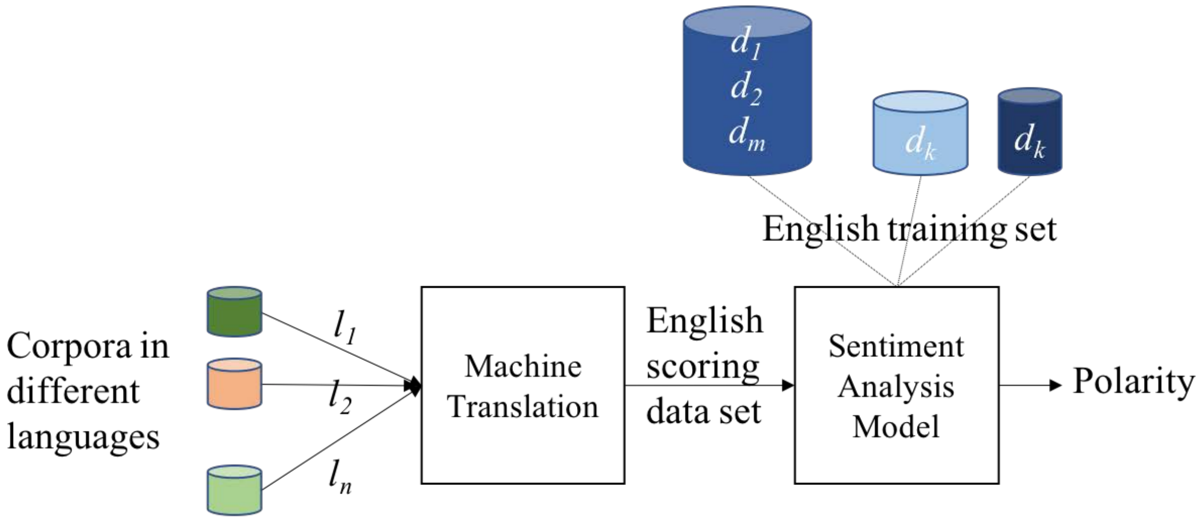


Figure 1. Multilingual Sentiment Analysis Approach

Deep learning approaches have been successful in many applications, ranging from computer vision to natural language processing (Alom et al. 2018). Recurrent neural networks (RNNs) including long short-term memory (LSTM) and gated recurrent units (GRUs) are subsets of deep learning algorithms in which the dependencies between tokens can be used by the model. These models can also be used with variable-length input vectors, making them suitable for text input. LSTM and GRU models allow operations of sequences of vectors over time and have the capability to remember previous information (Alom et al. 2018).

RNNs have been found useful for several natural language processing tasks, including language modeling, text classification, and machine translation. An RNN can also use pretrained word embeddings (numeric vector representations of words that are trained on unlabeled data) without requiring hand-crafted features. Therefore, in this paper, we use an RNN architecture that takes text and pretrained word embeddings as inputs and generates a classification result. Word embeddings represent words as numeric vectors and capture semantic information. They are trained in an unsupervised fashion, making them useful for our task.

The sentiment analysis model that is trained on English reviews has three bidirectional LSTM layers, each with 50 neurons. The training phase takes pretrained word embeddings and reviews in textual format, and then predicts the polarity of the reviews. For this study, an embedding length of 100 is used (that is, each word is represented by a vector of length 100). The training phase is depicted in Figure 2.

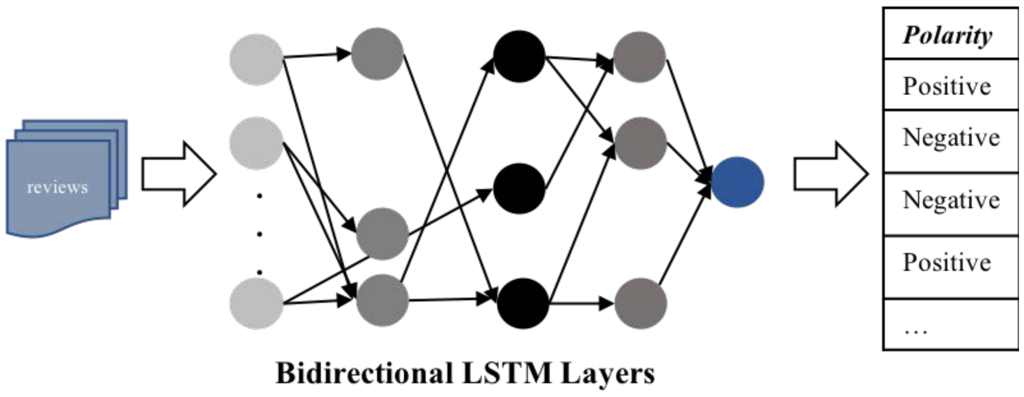


Figure 2. Training Sentiment Analysis Model That Uses LSTM

EXPERIMENTS

To evaluate the proposed approach for our multilingual sentiment analysis task, we conducted experiments. This section first presents the corpora that we used in this study and then shows the experimental results. Throughout our experiments, we used the SAS Deep Learning toolkit.

CORPORA

We used two corpora in this study, both of which are publicly available. The first corpus consists of English reviews, and the second corpus contains restaurant reviews in five different languages (Turkish, Spanish, Russian, Dutch, and Chinese). We focused on polarity detection in reviews; therefore, all data sets in this study have two class values (positive and negative).

For training our LSTM model, we used the data set provided by Kotzias et al. (2015). For evaluation of the multilingual approach, we used five languages. These data sets are part of the SemEval-2016 Challenge Task 5 (Pontiki et al. 2016).

EXPERIMENTAL RESULTS

For experimental results, we reported the majority baseline for each language, where the majority baseline corresponds to a model's accuracy if it always predicts the majority class in the data set. For example, if 60% of all reviews in the data set are positive and 40% are negative, the majority baseline would be 60% because a model that always predicts "positive"' will be 60% accurate and will make mistakes 40% of the time.

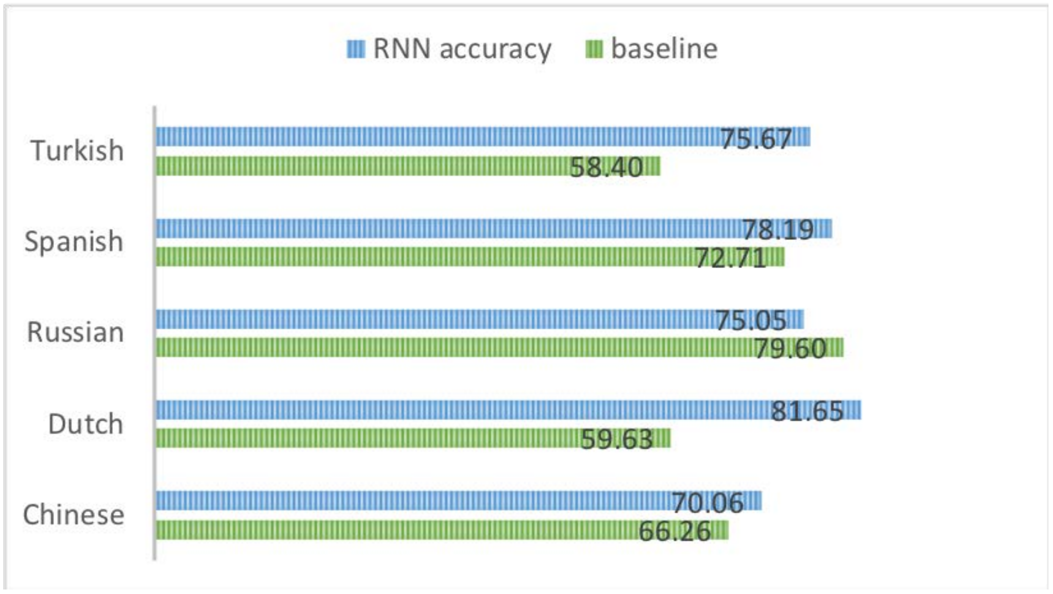


Figure 3. Test Set Accuracy Results for Different Languages

Experimental results on the test set are depicted in Figure 3. An RNN outperforms the baseline in four of the languages (Turkish, Spanish, Dutch, and Chinese). Note that the test set contains reviews from different domains, thus making it challenging for a single model to perform well on all of these languages.

To further analyze how well RNN performed for each language, we calculated the accuracies per class value. This analysis gave us two accuracy values for each language: one for the positive reviews, and one for the negative reviews. Figure 4 shows the accuracies of both positive and negative reviews for all five languages. The general trend is that accuracies of negative reviews are higher than accuracies of positive reviews. This can be explained by the imbalanced nature of the data set: because negative reviews are more frequently found in the data, it is easier for the model to learn from those negative reviews and therefore perform better on that class value.

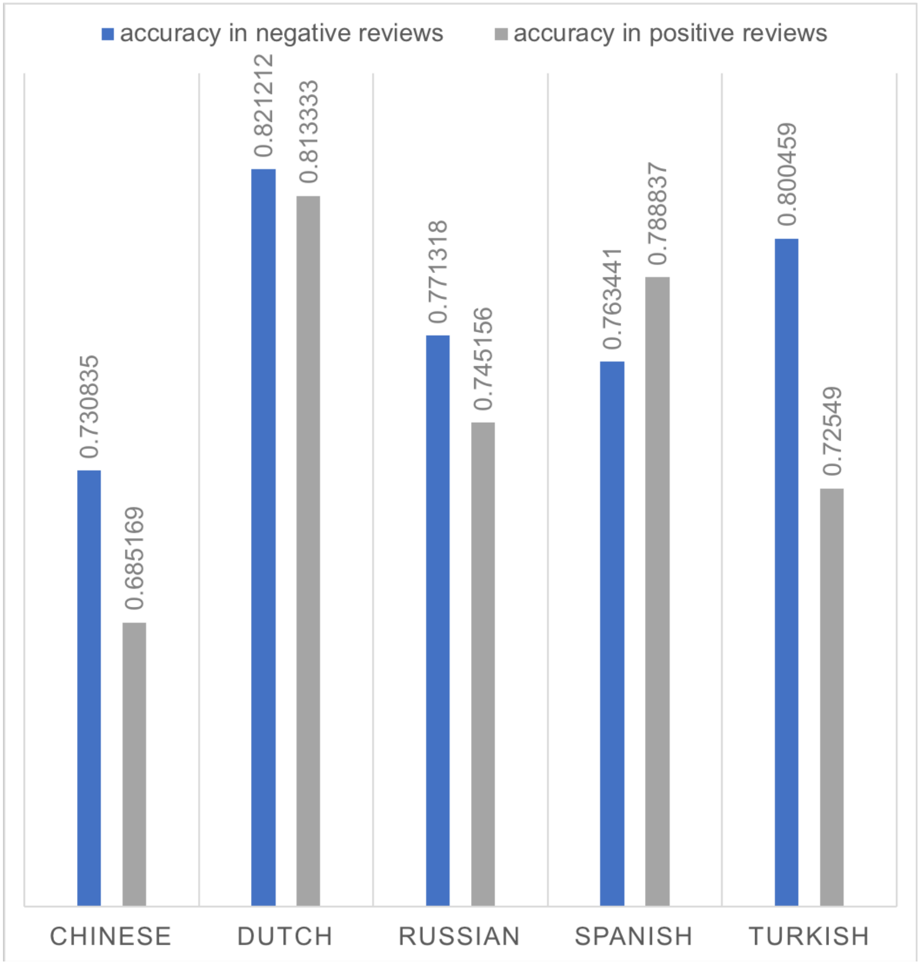


Figure 4. Accuracies of Positive and Negative Reviews

DISCUSSION

One of the crucial elements in using machine translation is to have highly accurate translations. We analyzed the effect of incorrect translations in our approach. To that end, we evaluated our model with an English corpus (Pontiki et al. 2016) to see its performance without any interference from machine translation errors.

Using the English data for testing, the model achieved 79.8% accuracy, where a majority baseline was 68.37%. In the test data, 84% of negative reviews and 77.84% of positive reviews were correctly classified.

Considering the improvements that were achieved by the RNN model over the majority baseline for both English and non-English reviews, we can draw the conclusion that our model is robust in handling multiple languages. Building separate models for each language requires both labeled and unlabeled data. Although having lots of labeled data in every language is the perfect case, it is unrealistic. Therefore, eliminating the resource requirement in this resource-constrained task is crucial. The fact that machine translation can be used in reusing models from different languages is promising for reducing the data requirements.

CONCLUSION

Building effective machine learning models for text requires data and other resources such as pretrained word embeddings and reusable lexicons. Unfortunately, most of these resources are not entirely transferable to different domains, tasks, or languages. Sentiment analysis is one such task that requires additional effort to transfer knowledge between languages.

In this paper, we studied the research question: Can we build reusable sentiment analysis models that can be used for making inferences in different languages without requiring separate models and resources for each language? To that end, we built a recurrent neural network model in the language that had largest amount of data available. During scoring, we used corpora from different domains in different languages and translated them to English to be able to classify sentiments by using the trained model. Experimental results showed that the proposed multilingual approach outperforms the baseline.

REFERENCES

Alom, M. D., T. M. Taha, C. Yakopcic, S. Westbert, P. Sidike, M. S. Nasrin, B. C. Van Esesn, et al. 2018. "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches." arXiv:1803.01164

Banea, C., R. Mihalcea, and J. Wiebe. 2008. "A Bootstrapping Method for Building Subjectivity Lexicons for Languages with Scarce Resources." 2008. *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. Vol. 8.

Inui, T., and M. Yamamoto. 2011. "Applying Sentiment-Oriented Sentence Filtering to Multilingual Review Classification." *Proceedings of the Workshop on Sentiment Analysis Where AI Meets Psychology (SAAIP)*. IJCNLP 2011. 51–58.

Kotzias, D., M. Denil, N. de Freitas, and P. Smyth. 2015. "From Group to Individual Labels Using Deep Features." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ortigosa, A., J. M. Martín, and R. M. Carro. 2014. "Sentiment Analysis in Facebook and Its Application to e-Learning." *Computers in Human Behavior* 31: 527–541.

Pontiki, M., D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar. M. Al-Smadi, M. Al-Ayoub, et al. 2016. "SemEval-2016 Task 5: Aspect Based Sentiment Analysis." *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*.

Steinberger, J., M. Ebrahim, M. Ehrmann, A. Hurriyetoglu, M. Kabadjov, P. Lenkova, R. Steinberger, et al. 2012. "Creating Sentiment Dictionaries via Triangulation." *Decision Support Systems* 53.4: 689–694.

Taboada, M., J. Brooke, M. Tofiloski, K. Voll, and M. Stede. 2011. "Lexicon-Based Methods for Sentiment Analysis." *Computational Linguistics* 37.2: 267–307.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Ethem Can
SAS
ethem.can@sas.com

Aysu Ezen-Can
SAS
aysu.ezencan@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Paper SAS4429-2020

NLP with BERT: Sentiment Analysis Using SAS® Deep Learning and DLPy

Doug Cairns and Xiangxiang Meng, SAS Institute Inc.

ABSTRACT

A revolution is taking place in natural language processing (NLP) as a result of two ideas. The first idea is that pretraining a deep neural network as a language model is a good starting point for a range of NLP tasks. These networks can be augmented (layers can be added or dropped) and then fine-tuned with transfer learning for specific NLP tasks. The second idea involves a paradigm shift away from traditional recurrent neural networks (RNNs) and toward deep neural networks based on Transformer building blocks. One architecture that embodies these ideas is Bidirectional Encoder Representations from Transformers (BERT). BERT and its variants have been at or near the top of the leaderboard for many traditional NLP tasks, such as the general language understanding evaluation (GLUE) benchmarks. This paper provides an overview of BERT and shows how you can create your own BERT model by using SAS® Deep Learning and the SAS DLPy Python package. It illustrates the effectiveness of BERT by performing sentiment analysis on unstructured product reviews submitted to Amazon.

INTRODUCTION

Providing a computer-based analog for the conceptual and syntactic processing that occurs in the human brain for spoken or written communication has proven extremely challenging. As a simple example, consider the abstract for this (or any) technical paper. If well written, it should be a concise summary of what you will learn from reading the paper. As a reader, you expect to see some or all of the following:

- Technical context and/or problem
- Key contribution(s)
- Salient result(s)

If you were tasked to create a computer-based tool for summarizing papers, how would you translate your expectations as a reader into an implementable algorithm? This is the type of problem that the field of natural language processing (NLP) addresses. NLP encompasses a wide variety of issues in both spoken and written communication. The field is quite active, because many NLP problems do not have solutions that approach human performance.

Historically, NLP practitioners focused on solutions with problem-specific, handcrafted features that relied on expert knowledge. There was a shift starting in the early 2000s (Bengio et al. 2003) to data-driven, neural network–based solutions that learned features automatically. During this shift, a key idea emerged: training a neural network to function as a language model is a good foundation for solving a range of NLP problems (Collobert et al. 2011). This neural network could either provide context-sensitive features to augment a task-specific solution (Peters et al. 2018) or be fine-tuned to solve a specific NLP problem (Radford 2018; Devlin et al. 2018). Both approaches were extremely successful and led to speculation (Ruder 2018) that an NLP "ImageNet" moment was at hand. The sense of the speculation was that neural network–based NLP solutions were approaching or exceeding human performance, as they had in the field of image processing. The neural network advances in image processing were inspired by the ImageNet Large Scale Visual Recognition Challenge (image-net.org 2012); hence the notion of an "ImageNet" moment.

Until recently, most neural network–based NLP approaches focused on recurrent neural networks (RNNs). Unlike other types of neural networks, RNNs consider data ordering, so an RNN is well suited for text or audio data, where order matters. For each element in a sequence, the output of an RNN depends on the current element as well as state information. The state information is a function of the sequence element(s) previously observed and is updated for each new element. This enables the RNN to "remember" and thus learn how sequential data evolve over time. The state calculation also makes an RNN difficult to efficiently implement, so training can be extremely time-consuming.

In 2017, the dominance of the RNN approach was challenged by the Transformer architecture (Vaswani et al. 2017). The Transformer is based on an attention mechanism. You can think of an attention mechanism as an adaptive weighting scheme. The output of an attention mechanism for sequence element n is a weighted sum of all the input sequence elements. The "adaptive" part refers to the fact that weights are trained for each sequence position. Attention (Transformer) differs from recurrence (RNN) in that all sequence elements are considered simultaneously. This approach has both performance and implementation advantages.

Bidirectional Encoder Representations from Transformers (BERT) combines language model pretraining and the Transformer architecture to achieve impressive performance on an array of NLP problems. Subsequent sections present an overview of BERT and a tutorial on how to build and train a BERT model using SAS Deep Learning actions and DLPy.

BERT OVERVIEW

This overview presents BERT from the perspective of an NLP practitioner—that is, someone primarily interested in taking a pretrained BERT model and using transfer learning to fine-tune it for a specific NLP problem. The key considerations for an NLP practitioner are the input data representation, the model architecture, and keys for successful transfer learning, all of which are discussed in the following sections.

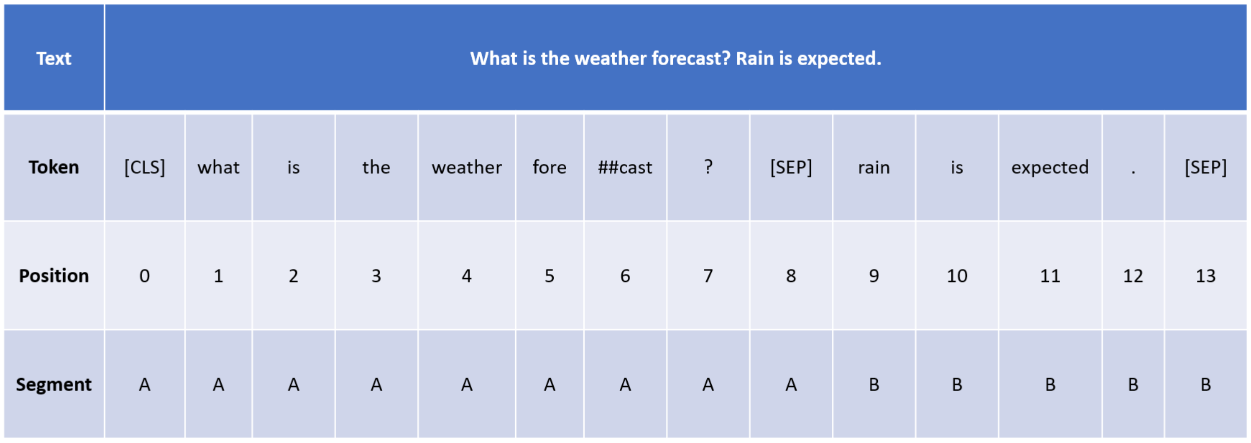
INPUT REPRESENTATION

Neural networks cannot operate directly on raw text data, so a standard practice in NLP is to tokenize the text (that is, split the text into meaningful phrase, word, or subword units) and then replace each token with a corresponding numeric embedding vector. BERT follows this standard practice but does so in a unique manner. There are three related representations required by BERT for any text string. The first is the tokenized version of the text. The second is the position of the token within the text string, which is something BERT inherits from the Transformer. The third is whether a given token belongs to the first sentence or the second sentence. This last representation makes sense only if you understand the BERT training objectives. A BERT model is trained to perform two simultaneous tasks:

- *Masked language model*. A fraction of tokens is masked (replaced by a special token), and then those tokens are predicted during training.
- *Next sentence prediction (NSP)*. Two sentences are combined, and a prediction is made as to whether the second sentence follows the first sentence.

The NSP task requires an indication of token/sentence association; hence the third representation. Both training objectives require special tokens ([CLS], [SEP], and [MASK]) that indicate classification, separation, and masking, respectively. In practice, this means that BERT text input is decomposed to three related values for each token. Figure 1 shows the three values for all tokens in this simple example:

"What is the weather forecast? Rain is expected."



Text	**What is the weather forecast? Rain is expected.**													
Token	[CLS]	what	is	the	weather	fore	##cast	?	[SEP]	rain	is	expected	.	[SEP]
Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Segment	A	A	A	A	A	A	A	A	A	B	B	B	B	B

Figure 1: BERT input for example

The example illustrates several important points. First, a classification token ([CLS]) begins every tokenized string, and separation tokens ([SEP]) conclude every sentence. Second, some words (such as *forecast*) can be split as part of the tokenization process. This is normal and follows the rules of the WordPiece model that BERT employs. Finally, you see that tokens in the first sentence are associated with segment A, while tokens in the second sentence are associated with segment B. In the case of a single sentence, all tokens are associated with segment A.

ARCHITECTURE

A BERT model has three main sections, as shown in Figure 2. The lowest layers make up the embedding section, which is composed of three separate embedding layers followed by an addition layer and a normalization layer. The next section is the Transformer encoder section, which typically consists of *N* encoder blocks connected sequentially (that is, output of encoder block 1 connects to input of encoder block 2, . . . , output of encoder block $N-1$ connects to input of encoder block *N*). Figure 2 shows $N=1$ for ease of illustration. The final section is customized for a specific task and can consist of one or more layers.

Embedding

The embedding section maps the three input values associated with each token in the tokenized text to a corresponding embedding vector. The embedding vectors are then summed and normalized (Ba, Kiros, and Hinton 2016). Here, the term *maps* refers to the process of using the token, position, or segment value as a key to extract the corresponding embedding vector from a dictionary or lookup table.

The token embedding layer maps token input values to a WordPiece embedding vector. The token embedding table/dictionary contains slightly more than 30,000 entries. The position embedding layer maps the position input values to a position embedding vector. Note that the position input value can be no greater than 512, so the position embedding table/dictionary contains 512 entries. The position value restriction also limits the length of the raw text string. The segment embedding layer maps the segment input value to one of two segment embedding vectors. All embedding vectors are of dimension *D*, where *D* is typically 768 or greater.

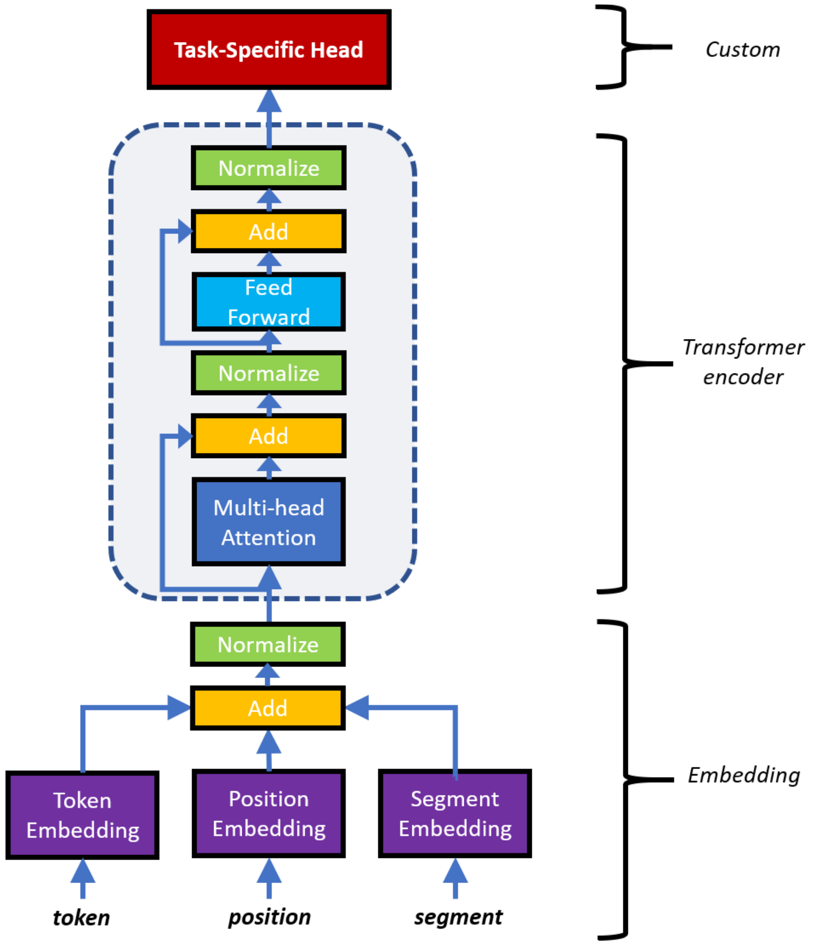


Figure 2: Simplified BERT model

Transformer Encoder

The encoder block consists of layers already encountered in the embedding section—namely, addition and normalization. There are also two composite layers, called feedforward and multi-head attention. The feedforward layer consists of two back-to-back fully connected (or dense) layers, where each input neuron is connected to all output neurons. Multi-head attention, shown in Figure 3, is more complex; there are three features of this composite layer to highlight.

First, you can see that multi-head attention requires three inputs, shown in the figure as $\mathbf{X}_\mathbf{Q}$, $\mathbf{X}_\mathbf{K}$, and $\mathbf{X}_\mathbf{V}$. These are matrices, where each row is a vector of dimension *D* that corresponds to a token from a tokenized text string. The first row represents the first token in the string, the second row represents the second token, and so on. In Transformer terminology, $\mathbf{X}_\mathbf{Q}$, $\mathbf{X}_\mathbf{K}$, and $\mathbf{X}_\mathbf{V}$ are the *query*, *key*, and *value* inputs, respectively. Whereas multi-head attention requires three inputs, Figure 2 shows only a single input to the encoder. This is because the BERT encoder uses self-attention (that is, $\mathbf{X}_\mathbf{Q} = \mathbf{X}_\mathbf{K} = \mathbf{X}_\mathbf{V}$).

Second, notice that there are multiple attention "heads." An attention head projects the *query*, *key*, and *value* inputs to independent lower-dimensional subspaces. This is followed by scaled dot-product attention. BERT builds on the Transformer, and the rationale for multiple heads is given in a comment in the original Transformer paper (Vaswani et al. 2017): "Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this."

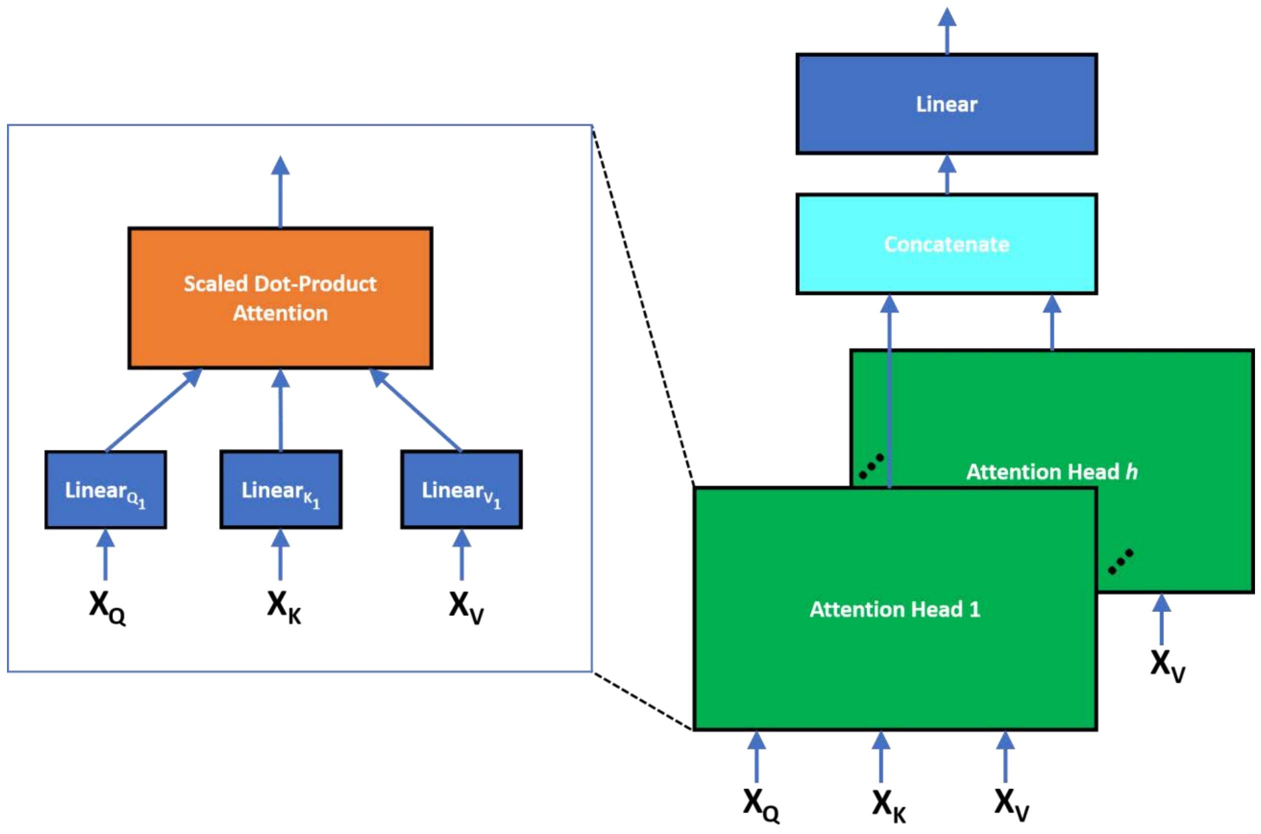


Figure 3: Multi-head attention

The final feature to highlight is the scaled dot-product attention mechanism. To gain some insight, consider the defining attention equation

$$Attention(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \frac{softmax\left(\mathbf{Q}_i\mathbf{K}_i^T\right)}{\sqrt{d}}\mathbf{V}_i$$

where the *softmax* operator applies to each row of the matrix product $\mathbf{Q}_i\mathbf{K}_i^T$. The scaling term d is the dimension of the subspace. The subspace dimension is equal to D/H, where H is the number of attention heads. For a vector $\mathbf{x}$, the *softmax* operator returns a vector, the *m*th element of that vector, given by

$$softmax(x_m) = \frac{e^{-x_m}}{\sum_j e^{-x_j}}$$

Now consider what the attention equation computes. For the *i*th attention head, matrices $\mathbf{Q}_i$, $\mathbf{K}_i$, and $\mathbf{V}_i$ are d-dimension subspace projections of matrices $\mathbf{X}_\mathbf{Q}$, $\mathbf{X}_\mathbf{K}$, and $\mathbf{X}_\mathbf{V}$:

$$\mathbf{Q}_i = \mathbf{X}_\mathbf{Q}\mathbf{W}_{\mathbf{Q}_i}$$
$$\mathbf{K}_i = \mathbf{X}_\mathbf{K}\mathbf{W}_{\mathbf{K}_i}$$
$$\mathbf{V}_i = \mathbf{X}_\mathbf{V}\mathbf{W}_{\mathbf{V}_i}$$

Focusing on $softmax(\mathbf{Q}_i\mathbf{K}_i^T)\mathbf{V}_i$, notice that row *m* of matrix $\mathbf{Q}_i\mathbf{K}_i^T$ is the cross-correlation of the *query* subspace token at position *m* with all *key* subspace tokens for a given tokenized string. After the *softmax* operation, the row *m* cross-correlation values become a set of weights for combining all the *value* subspace tokens to create a new subspace token *m* representation. Here you see the concept of attention at work: the new token *m* is most influenced by (pays the most attention to) those *value* subspace tokens with large weights and is least influenced by (generally ignores) those *value* subspace tokens with small weights.

Scaled dot-product attention concludes each attention head, but there is a final step that allows each head in the encoder to contribute to the new *D*-dimensional representation for each token. The encoder applies a fully connected layer to the concatenated output of all attention heads, mixing the token representations from the *value* subspaces for each head.

Task-Specific Layer(s)

The custom section of the BERT model is tailored to a task, so a general overview is difficult. However, because classification-type scenarios arise in many NLP situations, a simple example is possible. For classification scenarios, the task-specific head is just a fully connected layer, in which the number of output neurons is equal to the number of classes. If the classification task is something like sentiment analysis, then the classification decision uses the output of the fully connected layer associated with the [CLS] token. If the classification task is token-specific (such as named entity recognition), then the classification decision uses the output of the fully connected layer associated with the token(s) in question.

TRANSFER LEARNING

Transfer learning describes the following process:

1. Obtaining an appropriate model with pretrained parameters
2. Removing layer(s) specific to the original model objective
3. Adding layer(s) specific to the new model objective
4. Performing fine-tuning training to optimize model parameters

In general, steps 1 and 4 are the key steps in the process. There are multiple sources for pretrained BERT models, and any of them might be suitable for your application. One popular source is the HuggingFace *transformers* GitHub repository. There you will find many BERT pretrained models for a variety of scenarios (English-only, multilingual, and so on). You should choose the model that best matches your scenario. After selecting a BERT model, you must then fine-tune it with data specific to your problem. Like the original model training, the fine-tuning training requires the selection of an optimization algorithm along with associated training hyperparameters. This can be a time-consuming exercise, but fortunately, Devlin et al. (2018) provide some helpful suggestions. The Adam optimization algorithm (Kingma and Ba 2015) is recommended with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Also recommended are the hyperparameter settings shown in Table 1.

Hyperparameter	Setting
Dropout	0.1
Batch size	8, 16, 32
Learning rate	$2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}$
Number of epochs	2, 3, 4

Table 1: Recommended hyperparameter settings for fine-tuning

TUTORIAL: FINE-TUNING A BERT MODEL

You can use SAS Deep Learning and the SAS DLPy Python package to build and fine-tune a BERT model. To illustrate this process, consider performing sentiment analysis on an Amazon review data set with a BERT classification model. This tutorial example walks through the five steps you must perform and concludes with an evaluation of the trained model.

PREREQUISITES

This example assumes the following:

- The Amazon Fine Food Reviews data set from the Kaggle competition website is available on the client computer.
- You have a working understanding of Python.
- SAS Scripting Wrapper for Analytics Transfer (SWAT) is available on the client computer (clone, fork, or install from here).
- SAS DLPy is available on the client computer (clone, fork, or install from here).
- You have a working Python environment on the client computer that includes the *transformers* package from the HuggingFace GitHub repository and PyTorch. See the recommendations here for setting up a suitable Anaconda environment.
- A SAS® Viya® 3.5 server is set up and running.
- An active Viya session is running in the Python environment.

Note that SAS Viya is a client-server architecture, and there are references to both client and server in the preceding list. For the purposes of this tutorial, consider the Viya client to be a desktop PC and the Viya server to be a separate computer.

STEP 1: CREATE BERT CLASSIFICATION MODEL

The first step is to create a DLPy model object that encapsulates the BERT classification model. Start by defining the BERT cache directory, and then instantiate an object of the *BERT_Model* class. The *BERT_Model* object looks in the cache directory for BERT model definition and parameter information. If this information is absent, it will be downloaded from the HuggingFace repository:

```
from dlpy.transformers import BERT_Model

cache_dir = ‘path/to/your/cache-dir’

bert = BERT_Model(viya_conn,
                  cache_dir,
                  ‘bert-base-uncased’,
                  2,
                  num_hidden_layers = 12,
                  max_seq_len = 256,
                  verbose = True)
```

Note the *viya_conn* variable. This is the SWAT connection to the active Viya session referred to earlier.

STEP 2: PREPARE DATA

The second step is to prepare your data for training. Begin by reading the data from the Amazon Fine Food Reviews data set into a Pandas DataFrame:

```
import pandas as pd
reviews = pd.read_csv(‘name-of-your-amazon-review-file’,
                      header=0,
                      encoding=’utf-8’)
```

Then assign a numeric value to indicate positive or negative sentiment for each review. Since the number of stars associated with each review ranges from 1 to 5, filter out the neutral reviews (3 stars), and then assign a negative label to 1- and 2-star reviews and a positive label to 4- and 5-star reviews:

```
t_idx = reviews[“Score”] != 3
inputs = reviews[t_idx][“Text”].to_list()
targets = reviews[t_idx][“Score”].to_list()

for ii,val in enumerate(targets):
    inputs[ii] = inputs[ii].replace(“<br />”,””)
    if (val == 1) or (val == 2):   # negative reviews
        targets[ii] = 1
    elif (val == 4) or (val == 5): # positive reviews
        targets[ii] = 2
```

Finally, import the data preparation helper function. Invoking the helper function tokenizes the review data and creates the three input values (token, position, segment) associated with each token as well as the sentiment target for each review. The helper function also automatically creates a Viya table or tables containing the prepared data. The following invocation splits the reviews into a training set that has 80% of the overall data and a testing set that contains the remaining data:

```
from dlpy.transfomers.bert_utils import bert_prepare_data

num_tgt_var, train, test = bert_prepare_data(viya_conn,
                                             bert.get_tokenizer(),
                                             input_a=inputs,
                                             target=targets,
                                             train_fraction=0.8,
                           segment_vocab_size=bert.get_segment_size(),
                           classification_problem=bert.get_problem_type(),
                                             verbose=True)
```

STEP 3: CREATE SAS VIYA BERT MODEL

The third step is to create a SAS Deep Learning model that is the equivalent of the base BERT model plus a classification (fully connected) layer. The DLPy BERT model object provides a convenient function that performs this step for you:

```
bert.compile(num_target_var=num_tgt_var)
```

Note that this function does more than just create a BERT model. It also reads the trained parameters from the HuggingFace BERT model and saves them as an HDF5 file on the client computer. This file has a predefined structure that the SAS Deep Learning actions expect.

STEP 4: ATTACH MODEL PARAMETERS

The fourth step is to attach the trained model parameters stored in the HDF5 file to the BERT model. SAS Deep Learning actions read this HDF5 file, so it must be accessible by the Viya server. Because the client computer is assumed to be separate from the server computer, copy or move the HDF5 file to a location where it is visible to the server:

```
import os
from shutil import copyfile

server_dir = ‘path/to/your/server-directory’

copyfile(os.path.join(cache_dir,’bert-base-uncased.kerasmodel.h5'),
         os.path.join(server_dir,’bert-base-uncased.kerasmodel.h5'))
```

This example assumes that even though the client and server computers are distinct, they share a common file system. If that weren’t true, then some other means of moving the file from the client to the server (such as FTP) would be required. When the file has been successfully copied or moved, invoke the *load_weights()* function exposed by the DLPy BERT model object to attach parameters:

```
bert.load_weights(server_dir+'/bert-base-uncased.kerasmodel.h5',
                  num_target_var=num_tgt_var,
                  freeze_base_model=False)
```

The parameter *freeze_base_model* controls the training of the BERT model. It is set to False here; this allows training of all layers of the new BERT model. If you set it to True, then only the final classification layer could be trained. In that case, all other model parameters would be fixed.

STEP 5: FINE-TUNE BERT CLASSIFICATION MODEL

The final step is to perform fine-tuning training. Recall the fine-tuning recommendations provided by Devlin et al. (2018). The Adam optimizer and a default set of hyperparameters are defined for you when you invoke the *load_weights()* function in step 4. If the defaults are acceptable, you can invoke the *fit()* function exposed by the DLPy BERT model object. If you want to override one or more of the defaults, you can call the *set_optimizer_parameters()* function before invoking *fit()*, as follows:

```
bert.set_optimizer_parameters(learning_rate=2e-5)

bert.fit(train,
         data_specs= bert.get_data_spec(num_tgt_var),
         optimizer=bert.get_optimizer(),
         text_parms=bert.get_text_parameters(),
         seed=12345,
         n_threads=32))
```

When the *fit()* function finishes executing, your BERT model is fine-tuned for sentiment analysis.

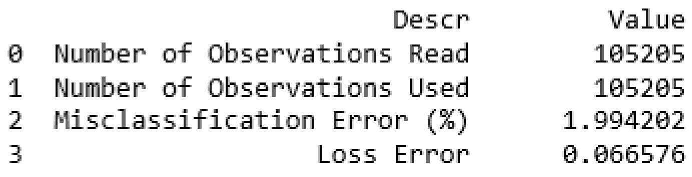
MODEL EVALUATION

You can now evaluate your fine-tuned sentiment analysis model by using the test data set. The *predict()* function exposed by the DLPy BERT model object provides a convenient method for this evaluation:

```
res = bert.predict(test,
                          text_parms=bert.get_text_parameters())


print(res['ScoreInfo'])
```

The results of the print function are shown in Figure 4. You can see that after three epochs of fine-tuning, the model achieves slightly better than 98.0% accuracy on the test data.



```
                            Descr       Value
0  Number of Observations Read        105205
1  Number of Observations Used        105205
2  Misclassification Error (%)      1.994202
3                      Loss Error    0.066576
```

Figure 4: Evaluation results

CONCLUSION

The field of natural language processing is undergoing a revolution, thanks to the ideas of language model pretraining and attention. BERT brings these concepts together to enable a powerful paradigm based on Transformer building blocks: simple fine-tuning using transfer learning provides state-of-the-art performance in many NLP tasks. This paper provides an NLP practitioner with an overview of key aspects of BERT and shows how to fine-tune and evaluate a BERT sentiment analysis model using SAS Deep Learning and the SAS DLPy Python package.

REFERENCES

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). "Layer Normalization." *arXiv:1607.06450v1*, 1–14.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). "A Neural Probabilistic Language Model." *Journal of Machine Learning Research* 3:1137–1155.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). "Natural Language Processing (Almost) from Scratch." *Journal of Machine Learning Research* 12:2493–2537.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *arXiv:1810.04805v1*, 1–14.

Image-net.org. (2012). "ImageNet Large Scale Visual Recognition Challenge." Accessed December 9, 2019. http://image-net.org/challenges/LSVRC/.

Kingma, D. P., and Ba, J. (2015). "Adam: A Method for Stochastic Optimization." 3rd International Conference for Learning Representations. San Diego: ICLR.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). "Deep Contextualized Word Representations." *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics,* 2227–2237. New Orleans: Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). "Improving Language Understanding with Unsupervised Learning." *OpenAI Technical Report*.

Ruder, S. (2018). "NLP's ImageNet Moment Has Arrived." Accessed December 9, 2019, https://ruder.io/nlp-imagenet.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). "Attention Is All You Need." *Advances in Neural Information Processing Systems 30*. NIPS 2017, Long Beach, CA.

RECOMMENDED READING

- *The Illustrated Transformer* (blog)
- *BERT for Dummies—Step by Step Tutorial* (blog)
- *HuggingFace Quickstart* (Transformers documentation)

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Doug Cairns
doug.cairns@sas.com

Xiangxiang Meng
xiangxiang.meng@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Paper SAS4434-2020

Sound Insights: A Pipeline for Information Extraction from Audio Files

Dr. Biljana Belamarić Wilsey and Xiaozhuo Cheng, SAS Institute Inc.

ABSTRACT

Audio files, like other unstructured data, present special challenges for analytics but also an opportunity to discover valuable new insights. For example, technical support or call center recordings can be used for quickly prioritizing product or service improvements based on the voice of the customer. Similarly, audio portions of video recordings can be mined for common topics and widespread concerns. To uncover the value hidden in audio files, you can use a pipeline that starts with the speech-to-text capabilities of SAS® Visual Data Mining and Machine Learning and continues with analysis of unstructured text using SAS® Visual Text Analytics software. This pipeline can be illustrated with data from the Big Ideas talk series at SAS, which gives employees the opportunity to share their ideas in short, TED Talk–type presentations that are recorded on video. If you ever wondered what SAS employees are thinking about when they're not thinking of ways to make SAS products better, the answers lie in a pipeline for information extraction from audio files. You can use this versatile pipeline to discover sound insights from your own audio data.

INTRODUCTION

It is a commonly accepted idea that in the modern world of big data, the most frequent type of data is unstructured: rich media, including video and audio; free-form text, including medical notes and document collections, and so on. While some challenges, such as density and scale, are common to both structured and unstructured data, unstructured data presents additional unique challenges for analysis. One of these challenges arises because the unstructured data first needs to be converted to structured data (Bagga, 2013) and there are an almost infinite number of ways to do so, depending on the business question that is being asked of that data. But for those who are up to the task, unstructured data hides a wealth of insights.

Focusing specifically on audio data, the business value of insights from audio files is recognized by most industry analysts, who predict that the global speech and voice recognition market will reach $26B by 2025 (Marketwatch, 2019) and the speech-to-text application program interface market will reach $4.1B by 2025 (ReportLinker, 2019). This impact was foreshadowed by Donna Fluss of vendor-independent consulting firm DMG over a decade ago when she wrote: "When used properly and accompanied by best practices, speech analytics typically pays for itself in three to nine months" (Fluss, 2007). Using audio as a customer engagement channel and analytics tools to derive insights about the "voice of the customer," companies can, for example, evaluate the effectiveness of marketing campaigns, understand the experiences and sentiment of their customers, grow revenue (Salta, 2018), retain customers to increase market share, and increase efficiency by focusing on customer-reported differentiators (Kaplan, 2014). But you get no return-on-investment by just collecting data; the value comes from deriving insights and making them actionable items (Sage, 2013).

This paper describes how you can extract those insights from audio files, using five components of the speech-to-text capabilities of SAS Visual Data Mining and Machine Learning and four nodes of analysis of unstructured text using SAS Visual Text Analytics. We demonstrate using this pipeline with data from the Big Ideas talk series at SAS, which gives employees the opportunity to share their ideas in short, TED Talk–type presentations that are recorded on video. However, the pipeline is equally applicable to contact and call center data, technical support and other conversational data, and even live streaming audio data (by connecting to SAS Event Stream Processing).

SCENARIO: BIG IDEAS

Since 2017, SAS has hosted a company-internal presentation series entitled Big Ideas. For each event, a dozen employees are selected as presenters from hundreds of applicants all over the globe. The audience consists of employees, but the event is also video-recorded, and those recordings are available on the company's internal site. The series supports the spirit of lifelong learning and provides a forum to share ideas, stories, passions, and to think about the potential to apply or achieve something spectacular. But because of the small cohort of presenters at each event, the selection process is very competitive.

Because the event is recorded on video but not transcribed, it presented an ideal use case for applying the speech-to-insights pipeline outlined previously. We focused on answering the question: "What are SAS employees passionate about outside of their daily jobs?" As extra motivation, we also wanted to use the findings for our own actionable insights: "What topics and terms should we include in order to write a winning application for the next Big Ideas event?"

TRANSFORMING AUDIO DATA INTO INSIGHTS WITH SAS

Because the source format was video, we used the open-source software FFmpeg to extract audio from the video files. The software converted the video to audio files in stereo WAV format (44.1 kHz, 16-bit stereo).

FROM AUDIO TO TEXT

The pipeline for transforming audio to text includes the following steps:

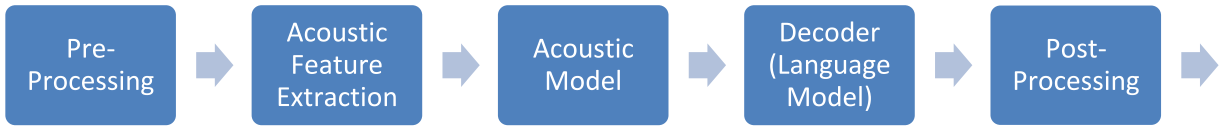


Figure 1. Speech-to-text Pipeline

For our project, the pre-processing step included these steps:

- converting the output from FFmpeg to 16 kHz 16-bit mono WAV, which is the standard supported input format for SAS speech-to-text pre-trained models
- segmentation of one long audio file into many short segments so that they could be processed in parallel and therefore faster

There are different methods for segmentation of long audio. For this project, we used a power-based algorithm to find pauses in the speech. Power is the absolute value of audio signals. Usually, an audio signal is higher in power when people speak than when there is silence. To segment long audio, we first chose a "low power threshold" that was based on the audio signal's power distribution. We considered values below this threshold to be silence. In addition, we specified a parameter called segment_len, which represented the maximum length in seconds that any audio segment could last.

The interaction between the low power threshold and segment_len for segmentation can be illustrated with the following example. We first specify segment_len of 30. In the first 30 seconds of the audio, we find the longest sub-sequence whose power values are all smaller than the low power threshold. We consider this sub-sequence as a pause in the speech, use its center point as the breakpoint to split the audio, and, starting from this breakpoint, move to the next 30-second period. We keep segmenting this way until the end of the audio file.

These segments are used as input for the acoustic feature extraction step, in which we break the segments down into much smaller units of analysis, known as time frames. For the current project, the time frames were 25 milliseconds in length. For each extracted time frame, specific acoustic features are extracted as vectors. The acoustic model then uses these features to convert sound into characters. For the current project, we chose the 40-dimension set of mel-frequency cepstral coefficients (MFCCs) features, which is one of the standards in the field.

The third step in the pipeline is the acoustic model. In speech recognition, acoustic models map the relationship between acoustic features and linguistic units (labels) that comprise utterances. Usually, different human (or natural) languages have different label sets. You can train your own model or use a pre-trained model for this step. SAS provides pre-trained acoustic models, which are available for download from the SAS Visual Data Mining and Machine Learning webpage as two versions: one for use with central processing units (CPUs) and one for graphical processing units (GPUs). For the current project, we used a pre-trained acoustic model that had these features:

- used MFCC features for training
- was character-based and English-specific, which means that the label set consisted of the alphabet in upper case, the apostrophe character, and the space character
- relied on recurrent neural networks (RNN) with multiple Long Short-Term Memory (LSTM) layers as well as fully connected layers

Using the feature vectors extracted in the previous step as input, the acoustic model computed and produced as output the probability distribution over the labels mentioned previously for each time frame.

The fourth component of the pipeline is the decoder, which is also known as a language model. Like the acoustic model, the language model also assigns probabilities. However, where the acoustic model assigns probabilities for character labels, the language model assigns probabilities for tokens or words, based on their sequence. For this project, we used a unidirectional trigram model, which assumes that the probability of the occurrence of a word depends on the previous two words in that sequence. Like acoustic models, SAS also provides pre-trained language models for download from the SAS Visual Data Mining and Machine Learning webpage. In this project, we trained our own language model in order to augment the vocabulary for the topics that we expected to be covered. Besides the large corpus of news used to train the general model in English, our training corpus also included several SAS internal blogs, in order to introduce vocabulary related to SAS. For this project, the decoder used beam search, which is a heuristic search algorithm that expands all possible next steps and keeps a fixed number (beam) of active candidates at each time step in order to find the best guess for the word. The beam size is a parameter the users can specify themselves. Usually, increasing the beam size improves the accuracy but decreases the speed of the system. The output from the decoder is a sequence of characters comprising words that correspond to the short audio segment that was the input into the acoustic feature extraction.

Because the audio input is segmented in the first step, the output from the language model is often in units smaller than a sentence. In some use cases of text mining, this approach might work well. For this project, we were interested in the topics that speakers addressed.

In this case, we found that short snippets of transcripts overemphasized frequent words[1], which generated less-useful topics. Therefore, in the post-processing step of the pipeline, we concatenated these short sequences of words in the same order as the original audio input to create longer transcripts of text. Each observation represented one Big Ideas speech. We found that this approach gave us more intuitive groupings of topics in SAS Visual Text Analytics.

The speech-to-text pipeline described above is supported in three different SAS products: SAS® Cloud Analytic Services (CAS), SAS® Event Stream Processing (ESP), and deep learning Python (DLPy). However, there are differences between the capabilities and parameters available in these three products. Users who prefer actions can convert speech to text step-by-step using the Audio, Deep Learning, and Language Model action sets. CAS also enables users to train the acoustic and language models by themselves. For users who have streaming data events, SAS ESP enables conversion of speech to text in real time. Finally, users who prefer Python can use DLPy’s end-to-end API that uses the file path of the WAV files as input and directly returns the transcript text.

FROM TEXT TO ACTIONABLE INSIGHTS

The speech transcripts, which were the output from the previous pipeline, are used as input into SAS Visual Text Analytics. To answer the research questions, we created a pipeline, which included the following nodes:

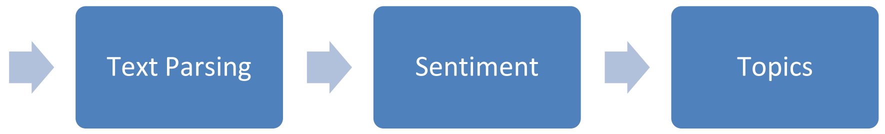


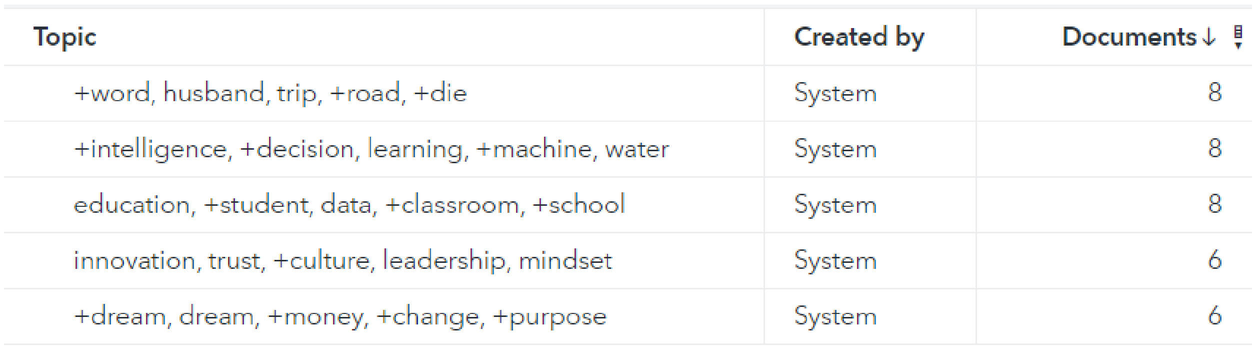
Figure 2. The Custom Pipeline in SAS Visual Text Analytics

We ran the entire pipeline and then started exploring the data by opening the Text Parsing node to find the most common terms and combinations of terms across all the speeches. The most common content terms included the terms “take,” “year,” “make,” “work,” and “people,” each of which occurred in more than 92% of the speeches. We were also interested in the context in which these terms occurred. We were able to easily find, for example, that the kinds of things the speakers discussed “taking” included risks and chances, breaks and time off, lessons, and so on. Looking at the term “make,” speakers talked about making choices and decisions, making things better, making a difference, making food such as casseroles and cookies, but also about making mistakes. As another example, the use of the term “year” was to provide the context of tenure at SAS, indicate age, and localize a point in time in the past when a particular event happened that the speakers shared. These contexts make sense when we think about the oratory strategy to tell personal stories to create a memorable emotional connection with the audience.

We also explored the data using the Topics node, which automatically detected common topics across the speeches and grouped them together. The pre-populated values (of 1) for the term and document density seemed to give the best results. As previously mentioned, we found that the Topics node autogenerated better topics with longer observations (such as one observation per speech) than with shorter snippets, which were broken up at pauses that speakers naturally made in intervals of 10 seconds or less.

[1] We estimated word frequency based on 11 words from the most frequent topics auto-generated by SAS Visual Text Analytics from short snippets and full speeches, using the word frequency from the 450-million-word corpus of Contemporary American English (on the website https://www.wordfrequency.info/free.asp?s=y). The frequency range of the words in the topics based on snippets was from 19 to 560 and the average was 199. The frequency range of the words in the topics based on the full speeches was from 157 to 3742 and the average was 941.

About two thirds of the speeches were grouped together automatically, out-of-the-box, into five topics, as illustrated in Figure 3. SAS Visual Text Analytics uses a term-document frequency matrix and reduces the dimensions with the singular value decomposition (SVD) method. Each topic is represented by the five most relevant terms. A plus sign in front of a term signifies that the software detected related forms, such as singular and plural of a noun or present and past tense of a verb, and automatically grouped them together. These related forms can be examined in the Text Parsing node.



Topic	Created by	Documents↓
+word, husband, trip, +road, +die	System	8
+intelligence, +decision, learning, +machine, water	System	8
education, +student, data, +classroom, +school	System	8
innovation, trust, +culture, leadership, mindset	System	6
+dream, dream, +money, +change, +purpose	System	6

Figure 3. Topics in SAS® Visual Text Analytics

The speeches grouped in the first topic included descriptions of road trips with families and friends as well as recounting of family relationships. One of these speeches, for example, described a very stressful road trip and concluded that one person's perception of a stressful trip can be another family member's memory of a wonderful time. Another speaker recounted family trips as she talked about coping with the loss of a family member. Because our pipeline included the Sentiment node ahead of the Topics node, the results of the node included a visualization of sentiment per topic. The emotionally difficult events described in the speeches in this topic contributed to a predominately negative sentiment for the topic, as seen in the third column in Figure 4.

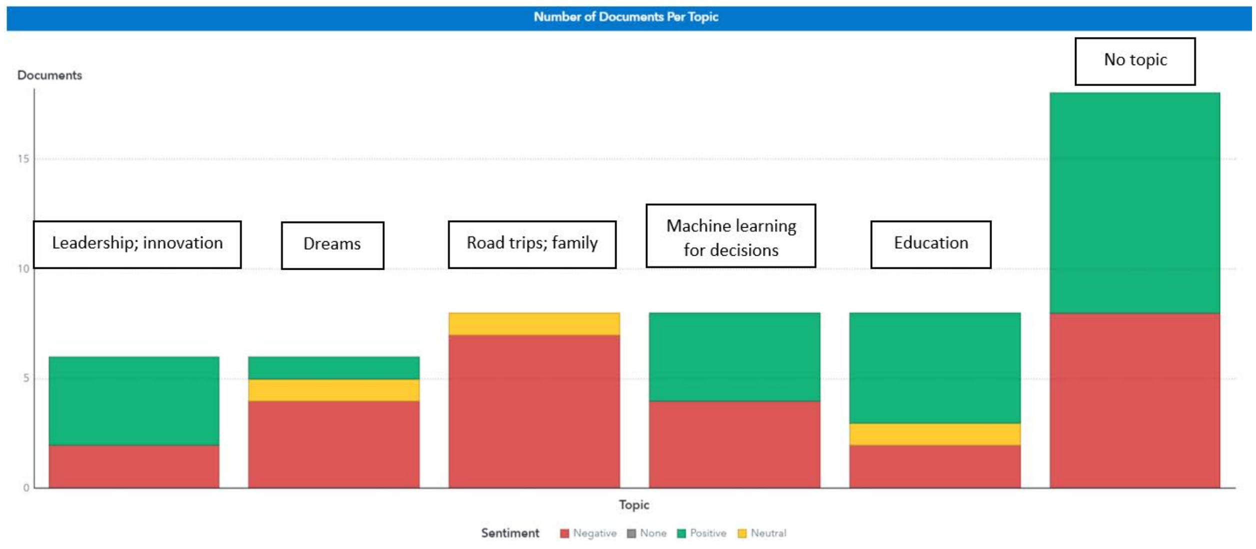


Figure 4. Sentiment of Topics in SAS Visual Text Analytics

The speeches grouped together in the second group centered on using machine learning (ML) and artificial intelligence (AI) for making better decisions. Two of them, for example, discussed how ML and AI can help solve the problems of human trafficking and shortages of clean drinking water. As seen in the fourth column of Figure 4, there were more speeches with positive sentiment on this topic than the previous topic. But there were also an equal number of speeches with negative sentiment.

The third topic centered on education, students, and data. For example, one speech discussed how we could potentially use data to better support students with disabilities and another talked about bringing data and analytics to students in classrooms. The sentiment of this topic was predominately positive, with one speech with neutral sentiment and two with negative sentiment.

The fourth topic combined ideas of leadership and innovation. Several speeches in this group focused on a culture of trust and showing vulnerability as part of successful leadership. Like the previous topic, the sentiment of most of the speeches in this topic was positive.

The fifth topic brought together speeches about dreams coming true, overcoming obstacles, persevering, and becoming successful. For example, one speech discussed the unique human abilities to dream and hope, whereas another provided tips for overcoming challenges and thriving through change. The sentiment of this group of speeches was predominately negative, probably because of the struggles inherent to achieving one's dreams.

The speeches that were not grouped in any of the topics above included unique speeches about, for example, "rocking a party" of AI geeks, creativity in work as in jazz, the power of numbers, the power of words, the value of taking chances, and others. There were more speeches with positive sentiment in this group than negative.

As we were browsing the speeches in the different topic areas, we realized that many talked about failures and struggles. Wanting to explore that area deeper, we added an additional node in the pipeline, the Concepts node (Figure 5).

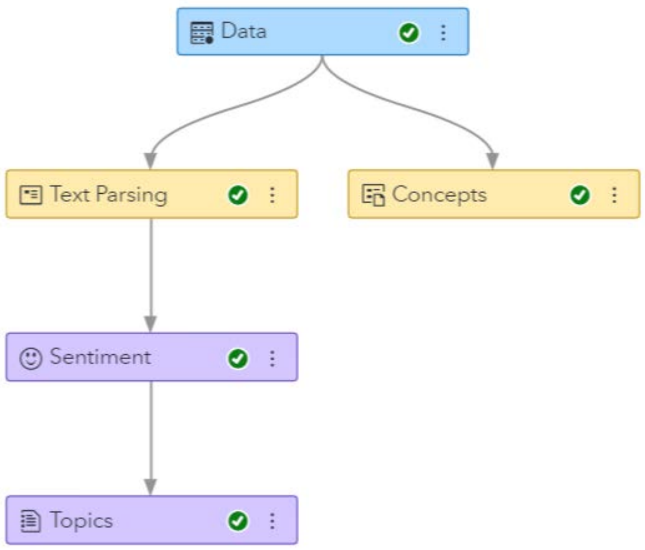


Figure 5. The final pipeline in SAS Visual Text Analytics

In the Concepts node, we created information extraction rules that captured terms, such as "problem," "difficult," "struggle," "disappoint", "hard", and "hardship." This part of the model building was a manual, iterative cycle of writing rules and examining output. We found that all but one of the speeches used these terms, with the exception being one speech about the courage to ask "what if" as the first step in encouraging innovation. We were also curious whether most of the talks also used positive terms such as "solution", "solve," "success" and "encourage." Therefore, we created a different concept with rules for extracting these terms. Less than two-thirds of the talks mentioned positive terms such as these. Therefore, the information extraction rules in the Concepts node showed that the speakers more commonly used terms describing struggling than succeeding.

CONCLUSION

In this paper, we explored how we can get insights from audio files of the Big Ideas talks at SAS. We used SAS Visual Data Mining and Machine Learning speech-to-text capabilities and SAS Visual Text Analytics to analyze unstructured data from speeches and derive insights for potentially applying to be selected for a future installment of the talk series.

Here are the data-driven insights we carry forward into writing our own abstracts for the Big Ideas series. Looking at frequently used content terms, we realized that the choice of words, including "take," "year," "make," "work," and "people," supported storytelling. Another insight the pipeline provided was that two-thirds of the talks were related to five main topic areas. Because these areas were popular in previous talks, chances are greater that proposing a future talk related to one of these topics might get accepted. In addition, nearly all talks specifically discussed a problem or a struggle, whether personal, societal, or global. Therefore, it seems very important that applicants for the talks mention a specific problem or hardship they are addressing, even more so than the solution.

This versatile speech-to-insights pipeline can be used to discover sound insights from your own audio data.

REFERENCES

Bagga, Simran. 2013. SAS Institute white paper. "Text Analytics: Unlocking the Value of Unstructured Data." https://www.sas.com/en/whitepapers/iia-text-analytics-unlocking-value-unstructured-data-108443.html.

Fluss, Donna. 2007. "Speech analytics converts call centers to profit centers." TechTarget. https://searchcustomerexperience.techtarget.com/news/1259565/Speech-analytics-converts-call-centers-to-profit-centers. Accessed on February 21, 2020.

Kaplan, Marcia. 2014. "Utilizing 'Voice of the Customer' for Competitive Advantage." Practical Ecommerce. https://www.practicalecommerce.com/Utilizing-Voice-of-the-Customer-for-Competitive-Advantage. Accessed February 21, 2020.

MarketWatch. 2019. "Speech and Voice Recognition Market Size, Growth, Opportunity, and Forecast to 2025." https://www.marketwatch.com/press-release/speech-and-voice-recognition-market-size-growth-opportunity-and-forecast-to-2025-2019-11-04. Accessed February 11, 2020.

Practical Cryptography. "Mel Frequency Cepstral Coefficient (MFCC) tutorial." http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. Accessed February 18, 2020.

NetApp. "What is Unstructured Data?" https://www.netapp.com/us/info/what-is-unstructured-data.aspx. Accessed February 11, 2020.

ReportLinker. 2019. "Speech-to-text API Market by Component, Application, Deployment Mode, Organization Size, Industry Vertical and Region – Global Forecast to 2024." https://www.reportlinker.com/p05826804/Speech-to-text-API-Market-by-Component-Application-Deployment-Mode-Organization-Size-Industry-Vertical-And-Region-Global-Forecast-to.html. Accessed February 11, 2020.

Sage, Adele. 2013. "Avoid the "All Listen and No Action" VoC Program Trap." Forrester. https://go.forrester.com/blogs/13-04-12-avoid_the_all_listen_and_no_action_voc_program_trap/. Accessed February 21, 2020.

Salta, Marissa. 2018. "CallTrackingMetrics Named in Forrester's Overview of AI-Fueled Speech Analytics Solutions." CallTrackingMetrics.

https://www.calltrackingmetrics.com/blog/press-releases/calltrackingmetrics-ai-fueled-speech-analytics-solutions. Accessed February 21, 2020.

"SAS Visual Text Analytics." https://www.sas.com/en_us/software/visual-text-analytics.html. Accessed February 11, 2020.

"SAS Visual Data Mining and Machine Learning." https://support.sas.com/en/software/visual-data-mining-and-machine-learning-support.html. Accessed February 11, 2020.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Biljana Belamarić Wilsey
SAS Institute
biljana.belamaricwilsey@sas.com

Xiaozhuo Cheng
SAS Institute
xiaozhuo.cheng@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Ready to take your SAS® and JMP® skills up a notch?

Be among the first to know about new books, special events, and exclusive discounts.
support.sas.com/newbooks

Share your expertise. Write a book with SAS.
support.sas.com/publish

sas.com/books
for additional books and resources.

sas®
THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2017 SAS Institute Inc. All rights reserved. M1588358 US.0217