Big Data Methods for Scraping Government Tax Revenue from the Web

Brian Dumbacher[1], Cavan Capps[1]
brian.dumbacher@census.gov, cavan.paul.capps@census.gov
[1]U.S. Census Bureau, 4600 Silver Hill Road, Washington, DC 20233

Abstract
The Quarterly Summary of State and Local Government Tax Revenue (QTax) is conducted by the U.S. Census Bureau to obtain data on tax revenue collections. Much of this data is publicly available on the web. Instead of responding via questionnaire, some respondents direct QTax analysts to their websites. An automated process for scraping data could reduce respondent burden and increase the timeliness of data products but is challenging to develop. There are thousands of government websites with little standardization, and most publications are in Portable Document Format (PDF), a file type not readily amenable to analysis. In this research, we focus on one part of the challenge and study how Big Data methods can be used to predict whether a previously unseen PDF contains content related to government tax revenue. Our methods use Python and natural language processing tools to extract, clean, and organize data from PDFs. A corpus of PDFs is compiled for machine learning purposes, and the performances of various classifiers are compared. Lastly, we discuss how these methods, in combination with a web crawler, can be used to automate the full process of scraping data.

Key Words: Big Data; Web scraping; Classification; Text analytics; PDF documents; Government units

1. Introduction

1.1 Motivation

The Quarterly Summary of State and Local Government Tax Revenue (QTax) is a sample survey conducted by the U.S. Census Bureau that collects data on tax revenue collections from state and local governments (U.S. Census Bureau, 2016). The taxes in scope to QTax include general sales and gross receipts tax, individual income tax, and corporate net income tax. Data on tax revenue collections are publicly available and can often be found on government websites. In fact, instead of responding via questionnaire, some respondents direct QTax analysts to their websites to obtain the data. Going directly to websites to obtain similar quality data has the potential to reduce respondent burden and increase the timeliness of QTax data products. Data found on government websites can also aid data review, imputation, and data verification.

An automated process for scraping tax revenue data from the web is ideal but challenging to develop. There are thousands of government websites but very little standardization in terms of website structure, data products, and publications. Also, a large majority of government publications are in Portable Document Format (PDF), a file type not readily amenable to analysis. In this paper, we focus on one part of the challenge and study how methods for unstructured data, text analytics, and classification can be used to predict whether a previously unseen PDF discovered through web crawling contains relevant tax revenue data.

Disclaimer: Any views expressed are those of the authors and not necessarily those of the U.S. Census Bureau.

1.2 Big Data

The methods used in this research belong to the realm of Big Data (Mayer-Schönberger, 2014). There is no agreed-upon definition of Big Data, but the term generally refers to data sources characterized by three V's:

1. **Volume**: a large number of observations or variables
2. **Velocity**: real-time or frequently generated data
3. **Variety**: various data formats and structures including no structure

Examples of Big Data include data scraped from the web, transaction data, and sensor data. Big Data are also known as "found," "organic," or "undesigned" data. These adjectives convey the notion that the data are being used for a purpose other than the purpose for which they were created. Therefore, when working with Big Data, it is important to consider how representative they are of the target population.

This research is an application of Big Data methods to QTax and is one of several projects being undertaken by the Census Bureau to explore the potential of leveraging Big Data sources to enhance the Census Bureau's economic programs. The goals of this overall Big Data effort include adding detail to data products, improving timeliness, reducing respondent burden, and improving efficiency and quality throughout the survey life cycle.

1.3 Outline of Paper

In Section 2, we provide background information on web scraping and the hierarchical structure of text in PDFs. Section 3 describes our algorithm for extracting text from PDFs and converting it to a TXT format more amenable to analysis. Next, in Section 4, we explain how we selected a sample of PDFs from state government websites for machine learning purposes. This sample is divided into training and test sets, which are used to evaluate classifiers that predict whether a previously unseen PDF contains content relevant to tax revenue collections. Section 5 describes these classifiers as well as features based on n-grams, which are sequences of words. In Section 6, results are given after fitting the classifiers to the training set and applying them to the test set. Section 7 describes limitations of the research. Finally, in Section 8, we discuss ideas for future research including how these classification methods, in combination with a web crawler, can be used to automate the full process of scraping data.

2. Background

2.1 Web Scraping

Web scraping is currently used by some European national statistical agencies to aid data collection. For instance, as described in Polidoro *et al.* (2015), the Italian National Institute of Statistics scrapes data on consumer electronics and airfares from the web for its harmonized index of consumer prices. The scraped data are lists of products and corresponding prices. This application involves hard-coded web scraping algorithms tailored to a handful of specific websites, so maintenance is required if the website or data structure changes. The methods are not fully automated and involve some human interaction. Polidoro *et al.* (2015) comment on the usefulness of a web scraping algorithm that completely replaces manual detection of product and price information.

Research has been conducted that tries to incorporate elements of learning and adaptability into web scraping. For example, the Big Data Task Force of the United Nations Economic Commission for Europe (UNECE) has explored developing a tool that crawls enterprise websites and scrapes data to create statistics on job vacancies (UNECE, 2016). This tool involves a classifier that predicts whether an advertisement is related to job vacancies. We learned about this work after conducting our own research and found that it captures much of the spirit of what we aim to accomplish.

High-quality open-source software is available for web scraping. Python and associated modules such as Beautiful Soup (Crummy, 2016) and NLTK: The Natural Language Toolkit (Bird, 2006) are popular tools for scraping and cleaning data. Mitchell (2015) provides an excellent overview of scraping the web with Python and gives many examples involving Beautiful Soup and NLTK. A nice framework for crawling and mapping websites is provided by two complementary programs from Apache: Nutch (Apache, 2014a) and Solr (Apache, 2014b). Nutch crawls the web and stores information about the crawl in a database, whereas Solr indexes and links the files discovered during the crawl.

2.2 Portable Document Format

Portable Document Format, or PDF, is a commonly used file format for presenting documents in a manner that does not depend on operating system (Adobe, 2016). Based on explorations of state and local government websites, it appears that a large majority of government publications are in PDF format. Text in PDFs is described in terms of a hierarchy of pages, textboxes, textlines, and individual characters. Understanding this hierarchy and the attributes at the various levels of the hierarchy are important for extracting text from a PDF properly.

Pages consist of textboxes, textboxes consist of textlines, and textlines consist of individual characters. In general, pages, textboxes, and textlines are assigned unique identification numbers that are ordered roughly according to how the text is read from top to bottom and from left to right. Textboxes, textlines, and individual characters have bounding boxes defined by two pairs of x- and y-coordinates. Figure 1 illustrates this hierarchy.

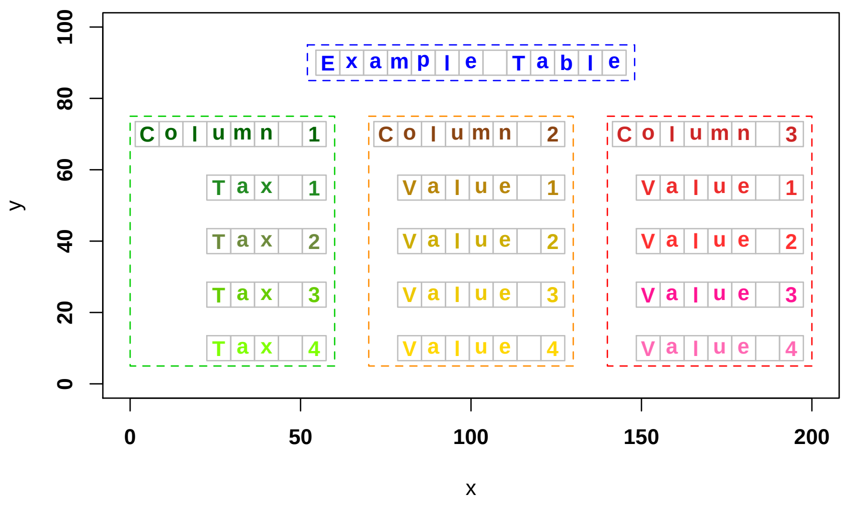


Figure 1: Textboxes are identified by dotted lines, and textlines are lines of text contained within textboxes. For example, the green textbox for Column 1 consists of five textlines. In turn, textlines consist of individual characters.

3. PDF-to-TXT Conversion

3.1 Algorithm

This section describes our PDF-to-TXT conversion algorithm for extracting text from PDFs and converting it to a TXT format amenable to text analysis and machine learning. Given a PDF, the steps are as follows:

1. Use the Python module PDFMiner (Shinyama, 2013) to extract the text from the PDF. PDFMiner extracts text represented as Unicode (Wikipedia, 2016) but does not recognize text drawn as images or text in scanned documents that would require optical character recognition (OCR). Also, PDFMiner is unable to extract text from password-encrypted PDFs. Based on results for state government websites presented in Section 4.1, about nine percent of all non-corrupt PDFs either are password-encrypted or contain text that can be recognized only using OCR.

2. Output the extracted text into Extensible Markup Language (XML) format. This format consists of a hierarchy of tags relating the pages, textboxes, textlines, and individual characters. Each character tag contains the corresponding character value, x- and y-coordinates of the bounding box, font name, and font size.

3. Use regular expressions (Mitchell, 2015, p. 22) to parse the XML file, and output individual character values and related information to a more easily readable intermediate TXT file. Convert punctuation and unusual characters such as letters with accents and foreign characters to white space.

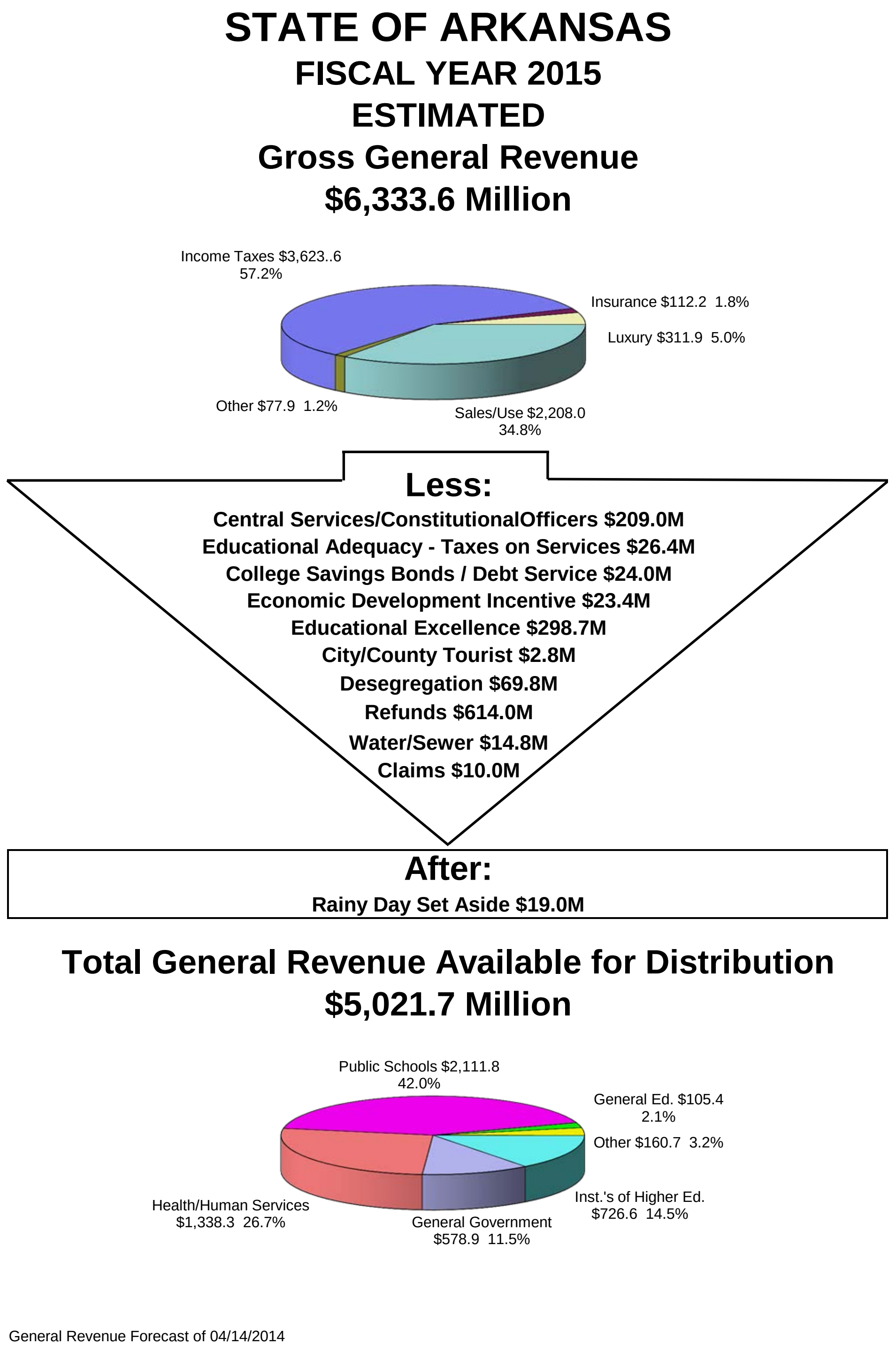
4. Based on page numbering, textbox numbering, textline numbering, and the x- and y-coordinates of bounding boxes, construct words character by character. Output these words to a separate TXT file.

5. Check each word against a comprehensive English dictionary and remove misspelled words and other errors. Also, remove common "stop" words such as articles, prepositions, and pronouns. The English dictionary we used is a TXT version of a Microsoft Excel dictionary created by Project Gutenberg (Ward, 2002). This dictionary contains about 355,000 words. Other English dictionaries that we considered using were not as comprehensive. For example, some did not contain plural nouns.

The entire conversion process takes a matter of seconds for most PDFs. The PDF-to-XML conversion is the most time-consuming step and can take an especially long time for lengthy PDFs such as comprehensive annual financial reports, which are issued by most state and local governments and are most beneficial to the Census Bureau's financial surveys such as QTax. Computer space may be an issue when converting a large number of PDFs because the resulting XML files tend to be quite large. Keeping the XML file during the conversion process is a good idea because it is the raw output from PDFMiner, and different data cleaning and stop word subroutines can be run on it. Parsing the XML file, creating the intermediate TXT file containing information about individual characters, and creating the final TXT file are relatively fast processes.

3.2 Example

Figure 2 on the next page is an example PDF from the Arkansas Department of Finance and Administration website. This PDF contains text and graphics related to estimated gross general revenue for fiscal year 2015.



STATE OF ARKANSAS
FISCAL YEAR 2015
ESTIMATED
Gross General Revenue
$6,333.6 Million

Income Taxes $3,623..6
57.2%

Insurance $112.2 1.8%

Luxury $311.9 5.0%

Other $77.9 1.2%

Sales/Use $2,208.0
34.8%

Less:

Central Services/ConstitutionalOfficers $209.0M
Educational Adequacy - Taxes on Services $26.4M
College Savings Bonds / Debt Service $24.0M
Economic Development Incentive $23.4M
Educational Excellence $298.7M
City/County Tourist $2.8M
Desegregation $69.8M
Refunds $614.0M
Water/Sewer $14.8M
Claims $10.0M

After:

Rainy Day Set Aside $19.0M

Total General Revenue Available for Distribution
$5,021.7 Million

Public Schools $2,111.8
42.0%

General Ed. $105.4
2.1%

Other $160.7 3.2%

Inst.'s of Higher Ed.
$726.6 14.5%

Health/Human Services
$1,338.3 26.7%

General Government
$578.9 11.5%

General Revenue Forecast of 04/14/2014

Figure 2: Example PDF document from the Arkansas Department of Finance and Administration website. Source: <http://www.dfa.arkansas.gov/offices/budget/Documents/fy15_gr_flowchart.pdf>

The conversion algorithm was run on the PDF in Figure 2. The following is part of the resulting XML file that starts at the first line and ends at the closing tag of the first textbox. The letters in this textbox spell “STATE OF ARKANSAS.”

```
<?xml version="1.0" encoding="utf-8" ?>
<pages>
<page id="1" bbox="0.000,0.000,612.000,792.000" rotate="0">
<textbox id="0" bbox="188.122,726.392,427.137,757.005">
<textline bbox="188.122,726.392,427.137,757.005">
<text font="Arial-BoldMT" bbox="188.122,726.392,202.770,757.005"
      size="30.612">S</text>
<text font="Arial-BoldMT" bbox="202.763,726.392,216.180,757.005"
      size="30.612">T</text>
<text font="Arial-BoldMT" bbox="216.202,726.392,232.058,757.005"
      size="30.612">A</text>
<text font="Arial-BoldMT" bbox="232.036,726.392,245.453,757.005"
      size="30.612">T</text>
<text font="Arial-BoldMT" bbox="245.475,726.392,260.122,757.005"
      size="30.612">E</text>
<text font="Arial-BoldMT" bbox="260.116,726.392,266.221,757.005"
      size="30.612"> </text>
<text font="Arial-BoldMT" bbox="266.243,726.392,283.328,757.005"
      size="30.612">O</text>
<text font="Arial-BoldMT" bbox="283.284,726.392,296.701,757.005"
      size="30.612">F</text>
<text font="Arial-BoldMT" bbox="296.723,726.392,302.828,757.005"
      size="30.612"> </text>
<text font="Arial-BoldMT" bbox="302.850,726.392,318.705,757.005"
      size="30.612">A</text>
<text font="Arial-BoldMT" bbox="318.683,726.392,334.538,757.005"
      size="30.612">R</text>
<text font="Arial-BoldMT" bbox="334.516,726.392,350.371,757.005"
      size="30.612">K</text>
<text font="Arial-BoldMT" bbox="350.350,726.392,366.205,757.005"
      size="30.612">A</text>
<text font="Arial-BoldMT" bbox="366.183,726.392,382.038,757.005"
      size="30.612">N</text>
<text font="Arial-BoldMT" bbox="382.016,726.392,396.663,757.005"
      size="30.612">S</text>
<text font="Arial-BoldMT" bbox="396.657,726.392,412.512,757.005"
      size="30.612">A</text>
<text font="Arial-BoldMT" bbox="412.490,726.392,427.137,757.005"
      size="30.612">S</text>
<text>
</text>
</textline>
</textbox>
```

The following is the corresponding part of the intermediate TXT file that contains information at the character level. Each record in this TXT file contains the page number, textbox number, textline number, the four coordinates of the bounding box, font size, font name, and character value.

```
1,0,1,188.122,726.392,202.770,757.005,30.612,"Arial-BoldMT","S"
1,0,1,202.763,726.392,216.180,757.005,30.612,"Arial-BoldMT","T"
1,0,1,216.202,726.392,232.058,757.005,30.612,"Arial-BoldMT","A"
1,0,1,232.036,726.392,245.453,757.005,30.612,"Arial-BoldMT","T"
1,0,1,245.475,726.392,260.122,757.005,30.612,"Arial-BoldMT","E"
1,0,1,260.116,726.392,266.221,757.005,30.612,"Arial-BoldMT"," "
1,0,1,266.243,726.392,283.328,757.005,30.612,"Arial-BoldMT","O"
1,0,1,283.284,726.392,296.701,757.005,30.612,"Arial-BoldMT","F"
1,0,1,296.723,726.392,302.828,757.005,30.612,"Arial-BoldMT"," "
1,0,1,302.850,726.392,318.705,757.005,30.612,"Arial-BoldMT","A"
1,0,1,318.683,726.392,334.538,757.005,30.612,"Arial-BoldMT","R"
1,0,1,334.516,726.392,350.371,757.005,30.612,"Arial-BoldMT","K"
1,0,1,350.350,726.392,366.205,757.005,30.612,"Arial-BoldMT","A"
1,0,1,366.183,726.392,382.038,757.005,30.612,"Arial-BoldMT","N"
1,0,1,382.016,726.392,396.663,757.005,30.612,"Arial-BoldMT","S"
1,0,1,396.657,726.392,412.512,757.005,30.612,"Arial-BoldMT","A"
1,0,1,412.490,726.392,427.137,757.005,30.612,"Arial-BoldMT","S"
```

Lastly, the following are the full contents of the final TXT file for use with text analysis and machine learning. The algorithm is able to extract the free text surrounding the pie charts as well as the text in the footnote.

```
state arkansas fiscal estimated gross general revenue
million income taxes insurance luxury sales less central
services educational adequacy taxes services college
savings bonds debt service economic development incentive
educational excellence city county tourist desegregation
refunds water sewer claims after rainy set aside total
general revenue available distribution million public
schools health human services general government general ed
inst higher ed general revenue forecast
```

4. PDFs for Machine Learning

4.1 Sample of PDFs

The goal of this research is to use machine learning methods to predict whether a previously unseen PDF discovered through web crawling contains relevant data on tax revenue collections. This is a two-class, or binary, classification problem (Tan, Steinbach, and Kumar, 2006, chaps. 4 and 5), and a corpus of PDFs with class labels already assigned is required for model fitting.

To this end, we compiled a list of state government websites thought to contain PDFs related to government finance. These websites included home pages of departments of revenue, taxation, and finance and home pages of state comptroller offices. We focused on state government websites because they are large and contain a wide variety of documents. For a list of these websites, see Appendix A.

Next, we used Nutch (Apache, 2014a) to crawl these websites to a depth of three, which means that all files within three links from the starting home page were discovered. Nutch stored the uniform resource locators of these files in a database. Crawling to a depth greater than three would have provided better coverage, but doing so required resources that were unavailable at the time. To prevent Nutch from crawling too far from the original home page, crawling was restricted to domains of the state government websites. In total, 59,578 PDFs were discovered.

From this universe of PDFs, we selected a simple random sample of size 6,000. These PDFs were manually curated and classified as positive (contains relevant data on tax revenue collections) or negative (does not contain relevant data). Some PDFs were found to be readable but password-encrypted and, thus, unable to be converted to the final TXT format. Other PDFs were unreadable simply because they were corrupt or could not be downloaded. Table 1 breaks down the sample according to these characteristics. There are 5,050 PDFs that could be converted to the final TXT format and used in the rest of the analysis.

Table 1: Breakdown of Sample of PDFs

	Class	Convertible	Inconvertible	Total
Readable	Positive	412	40	452
	Negative	4,638	463	5,101
Unreadable		–	447	447
	Total	5,050	950	6,000

4.2 Training and Test Sets

After the sample of 5,050 PDFs was classified and converted to final TXT format, it was randomly divided into a training set and a test set for use later with the holdout evaluation method (Tan, Steinbach, and Kumar, 2006, sec. 4.5.1). Two-thirds of the positive PDFs and two-thirds of the negative PDFs were randomly selected to form the training set. The remaining PDFs comprised the test set. Table 2 breaks down the training and test sets by class. In summary, 3,366 PDFs are used for training, and 1,684 PDFs are used for testing.

Table 2: Breakdown of Training and Test Sets

Class	Training Set	Test Set	Total
Positive	275	137	412
Negative	3,091	1,547	4,638
Total	3,366	1,684	5,050

5. Machine Learning Methods

5.1 Features

For each converted PDF in final TXT format, a vector of features is required that will help distinguish PDFs from the positive and negative classes. The features that we use are based on n-grams, which are sequences of n words appearing in the text. A 1-gram is a single word, a 2-gram is a pair of words, a 3-gram is a sequence of three words, and so on. As explained in Mitchell (2015, p. 109), n-grams are important in a natural language analysis, especially discriminating and frequently occurring n-grams.

In our application, we consider n-grams, where n = 1, 2, or 3. Each feature is a 0/1 indicator that indicates the presence of an n-gram in the PDF. Consider the Arkansas state government example in Section 3.2. Suppose this PDF is part of the training set on which classifiers are fit. The features consist of 0/1 indicators for all 1-grams, 2-grams, and 3-grams appearing in all PDFs in the training set. For the Arkansas PDF, the indicator for the 2-gram (state, arkansas) equals 1, but the indicator for the 2-gram (state, california), for example, equals 0.

5.2 Classifiers

The classification methods considered in this research are linear support vector classifiers (SVC) and Naïve Bayes (NB). The SVC classifier is given a thorough treatment in Tan, Steinbach, and Kumar (2006, sec 5.5) and James *et al.* (2013, chap. 9), whereas the NB classifier is described in Tan, Steinbach, and Kumar (2006, sec. 5.3.3). These two classification methods are considered in combination with seven sets of features, which, in turn, are based on combinations of 1-grams, 2-grams, and 3-grams. In the sections that follow, the sets of features are represented by shorthand notation. For example, (1,3) refers to 0/1 indicators for 1-grams and 3-grams, and SVC–(1,3) refers to the linear support vector classifier with this set of features. In total, there are 14 (= 2×7) classifiers under consideration.

The Python module NLTK is used to create the n-grams and features. The module Scikit-learn (Pedregosa *et al.*, 2015), which contains many tools for machine learning, is used in conjunction with NLTK to fit the SVC and NB classifiers. NLTK and Scikit-learn complement each other as the `nltk.classify.scikitlearn` package implements a wrapper around Scikit-learn classifiers. NLTK has its own implementations of some classifiers, and it was confirmed in testing that NLTK's NB classifier produces the same results as Scikit-learn's NB classifier for multinomial models. However, because the features are indicators of n-grams and not counts of n-grams, we decided it was more appropriate to use Scikit-learn's NB classifier for Bernoulli models.

6. Evaluation

6.1 Performance Measures

As mentioned in Section 4.2 on training and test sets, the holdout method is used to evaluate the performance of the classifiers under consideration (Tan, Steinbach, and Kumar, 2006, sec. 4.5.1). The classifiers are fit using the training set and then applied to the test set to predict class labels. The predicted class labels then are compared to the true class labels, and various performance measures can be calculated.

We follow the presentation in Tan, Steinbach, and Kumar (2006) to describe notation and performance measures for this binary classification problem. Table 3 is a confusion matrix that displays the number of true positives (TP), false negatives (FN), false positives (FP), and true negatives (TN) after making predictions for the observations in the test set. The total number of predictions is $M = TP + FN + FP + TN$.

Table 3: Confusion Matrix for a Binary Classification Problem

True	Predicted Class	
Class	Positive	Negative
Positive	TP	FN
Negative	FP	TN

One of the main performance measures is accuracy, AC, which is defined as the proportion of predictions that are correct,

$$AC = \frac{TP + TN}{TP + FN + FP + TN} = \frac{TP + TN}{M}.$$

For this application, it is more important to predict a positive PDF correctly than a negative PDF correctly. In other words, the cost of misclassifying a positive is worse than misclassifying a negative. A QTax analyst using a web scraping tool to find documents to aid data collection or data review can tolerate false positives, but there should be very few false negatives. In this regard, it is important to achieve a high true positive rate, TPR, also known as recall, which is defined as the proportion of positives that are classified correctly. The TPR is given by

$$TPR = \frac{TP}{TP + FN}.$$

Another commonly calculated quantity is the predictive positive rate, PPR, also known as precision, which is the proportion of positive predictions that are correct. The PPR is given by

$$PPR = \frac{TP}{TP + FP}.$$

A performance measure that balances the recall and precision is the F_1 score, which is given by

$$F_1 = \frac{2(TPR)(PPR)}{TPR + PPR} = \frac{2(TP)}{2(TP) + FP + FN}.$$

F_1 is the harmonic mean of TPR and PPR and takes on values between 0 and 1, where larger values are better.

6.2 Results

Table 4 presents values of the confusion matrix counts TP, FN, TN, and FP, and Table 5 presents values of the performance measures AC, TPR, PPR, and F_1 for all 14 classifiers.

Table 4: Confusion Matrix Counts

Method	Features	TP	FN	TN	FP
SVC	(1)	121	16	1,536	11
SVC	(2)	109	28	1,539	8
SVC	(3)	95	42	1,542	5
SVC	(1,2)	115	22	1,540	7
SVC	(1,3)	107	30	1,542	5
SVC	(2,3)	107	30	1,541	6
SVC	(1,2,3)	108	29	1,540	7
NB	(1)	51	86	1,392	155
NB	(2)	25	112	1,518	29
NB	(3)	6	131	1,542	5
NB	(1,2)	27	110	1,512	35
NB	(1,3)	9	128	1,540	7
NB	(2,3)	10	127	1,536	11
NB	(1,2,3)	14	123	1,535	12

Table 5: Performance Measures

Method	Features	AC	TPR	PPR	F_1
SVC	(1)	0.9840	0.8832	0.9167	0.8996
SVC	(2)	0.9786	0.7956	0.9316	0.8583
SVC	(3)	0.9721	0.6934	0.9500	0.8017
SVC	(1,2)	0.9828	0.8394	0.9426	0.8880
SVC	(1,3)	0.9792	0.7810	0.9554	0.8594
SVC	(2,3)	0.9786	0.7810	0.9469	0.8560
SVC	(1,2,3)	0.9786	0.7883	0.9391	0.8571
NB	(1)	0.8569	0.3723	0.2476	0.2974
NB	(2)	0.9163	0.1825	0.4630	0.2618
NB	(3)	0.9192	0.0438	0.5455	0.0811
NB	(1,2)	0.9139	0.1971	0.4355	0.2714
NB	(1,3)	0.9198	0.0657	0.5625	0.1177
NB	(2,3)	0.9181	0.0730	0.4762	0.1266
NB	(1,2,3)	0.9198	0.1022	0.5385	0.1718

The SVC classifiers perform better than the NB classifiers with respect to every performance measure. As shown by the low values of TP in Table 4 and the low values of TPR in Table 5, the NB classifiers do not predict positive PDFs accurately. The NB classifiers appear to have very good accuracy with AC around 0.91, but one must consider that a classifier that always classifies a PDF as negative would have an accuracy of 0.9186 (= 1547 / 1684) on this test set.

The highlighted rows in Tables 4 and 5 correspond to the two classifiers for each method with the largest values of F_1. Among the SVC classifiers, the SVC–(1) and SVC–(1,2) classifiers have the largest values of F_1 and AC. Additionally, SVC–(1) has the largest value of TPR. As mentioned in Section 6.1, this is a desirable property given the application to QTax. It is interesting to note that similar patterns among the performance measures exist for the NB–(1) and NB–(1,2) classifiers.

Table 6 lists the ten most important 1-grams, 2-grams, and 3-grams according to the NB–(1), NB–(2), and NB–(3) classifiers, respectively, in terms of discriminating positive and negative PDFs. Also reported is the positive-to-negative probability ratio for a PDF containing the n-gram. For example, under the NB–(2) model, a PDF containing the 2-gram (collections, month) is 227.8 times more likely to be positive than negative.

Table 6: Most Important 1-grams, 2-grams, and 3-grams According to Naïve Bayes Models

1-gram	Ratio	2-gram	Ratio
(constr)	93.4	(collections, month)	227.8
(devil)	85.9	(fuel, refunds)	153.1
(curr)	85.9	(refunds, net)	141.2
(riverboat)	82.9	(mining, utilities)	130.7
(addiction)	78.4	(enterprises, administrative)	130.7
(depict)	71.0	(statistical, report)	123.2
(boot)	71.0	(oil, severance)	115.8
(betting)	71.0	(add, due)	115.8
(dobson)	71.0	(activities, funds)	115.8
(defraying)	63.5	(title, fee)	115.8

3-gram	Ratio
(jefferson, county, county)	138.2
(tax, tobacco, tax)	138.2
(enterprises, administrative, support)	130.7
(companies, enterprises, administrative)	130.7
(mining, utilities, construction)	123.2
(sales, tax, income)	123.2
(remediation, services, educational)	123.2
(severance, tax, collections)	115.8
(county, miami, county)	115.8
(county, lake, county)	108.3

The results in Table 6 are for the NB models, but they highlight a matter relevant to any classifier based on the features we are considering. Even though we trained the classifiers on a representative sample of PDFs, using features based only on 1-grams may overemphasize the importance of words peculiar to the corpus of PDFs. For example, the seemingly odd 1-gram (devil) is one of the ten most important 1-grams. This is explained by the fact that the word "devil" appears in the names of multiple recreational areas in Arkansas. The features based on 2-grams seem to pick up on context and abstractions and could help models generalize better. We prefer the SVC–(1,2) classifier over the slightly better performing SVC–(1) classifier for this reason. The features based on 3-grams pick up on context, too, but not as strongly as the features based on 2-grams. Three of the most important 3-grams are related to specific counties: Jefferson County, Miami County, and Lake County. As with the 1-grams, using features based only on 3-grams may overemphasize the importance of words peculiar to the corpus of PDFs.

On an added note, we found the SVC method to perform much more efficiently than the NB method with respect to (1) model fitting and (2) the classification of new records. The efficiency of support vector methods is well known. For example, see James *et al.* (2013, chap. 9).

7. Limitations

The results in the previous section are based on a single sample of PDFs and are subject to sampling error. The sampling fraction is close to ten percent, which is fairly large, so we do not expect the results to change very much if we were to select additional samples. There is such a large difference in performance between the SVC and NB methods that we would expect the SVC classifiers to come out on top again. In machine learning applications, one typically does not take into account sampling variability associated with the training and test sets. One is more concerned with the representativeness of the two sets of the target population.

Also, we only considered two classification methods in this research. In the early stages of model fitting, we did investigate decision trees (Tan, Steinbach, and Kumar, 2006, sec. 4.3; James *et al.*, 2013, chap. 8) but found the methods to be slow given the number of features.

8. Future Research

The tool we have created is a combination of web crawler and classifier. Future research will involve fully integrating the two components into an automated process. Ideally, the tool would be pointed at a website and, with little or no analyst intervention, would then crawl the entire website, classify all PDFs it discovers, and maintain a history of actions. New PDFs could be fed back into the machine learning process to improve the performance of the classifier.

A good test of our methodology would be to apply the tool in its current form to the Census Bureau website and see whether QTax publications are found and classified as positive. For further testing, this tool also could be applied to less extensive websites of local governments that would allow for a deeper crawl.

For the classifiers in this research, features based only on 0/1 indicators of n-grams were considered, but many other features can be created such as counts of n-grams, the proportion of characters in the PDF that are numeric, the length of the PDF (in terms of the number of pages, textboxes, or textlines), and 0/1 indicators and counts of figures. Also, the number of features can be reduced by basing them on the most discriminating or most frequently occurring n-grams in the training set.

Once a PDF has been classified as positive, it would be useful to identify the actual data on tax revenue collections in the document. Such data would probably be found in tables and in close proximity to important n-grams. Table identification methods based on the density and distribution of characters in the PDF have been explored, and preliminary results show promise.

After identifying and scraping these data, putting them in a normalized data structure is the next Big Data challenge. In this regard, machine learning methods that map the non-standard data in PDFs from state and local government websites to the standard definitions in Census Bureau publications should be explored.

9. Acknowledgments

The authors would like to acknowledge Star Ying of the U.S. Department of Commerce for many helpful discussions, especially regarding research into extracting data from PDFs. Thanks also to Carma Hogue, Cheryl Lee, and Michael Kornbau of the U.S. Census Bureau for their comments and insight.

REFERENCES

Adobe. (2016). About Adobe PDF. <https://acrobat.adobe.com/us/en/why-adobe/about-adobe-pdf.html>. Accessed May 1, 2016.

The Apache Software Foundation. (2014a). Apache Nutch. <http://nutch.apache.org>. Accessed November 27, 2015.

The Apache Software Foundation. (2014b). Apache Solr. <http://lucene.apache.org/solr/>. Accessed November 27, 2015.

Bird, S. (2006). NLTK: The Natural Language Toolkit. *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Sydney, Australia: Association for Computational Linguistics, 69–72.

Crummy. (2016). Beautiful Soup. <https://www.crummy.com/software/BeautifulSoup/>. Accessed May 1, 2016.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. New York, NY: Springer.

Mayer-Schönberger, V. and Cukier, K. (2014). *Big Data*. New York, NY: Houghton Mifflin Harcourt.

Mitchell, R. (2015). *Web Scraping with Python: Collecting Data from the Modern Web*. Sebastopol, CA: O'Reilly Media, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Polidoro, F., Giannini, R., Lo Conte, R., Mosca, S., and Rossetti, F. (2015). Web Scraping Techniques to Collect Data on Consumer Electronics and Airfares for Italian HICP Compilation. *Statistical Journal of the IAOS*, *31*, 165–176.

Shinyama, Y. (2013). PDFMiner. <http://www.unixuser.org/∼euske/python/pdfminer/index.html>. Accessed November 23, 2015.

Tan, P.N., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining*. New York, NY: Pearson.

United Nations Economic Commission for Europe. (2016). Big Data – 2015 Project – Report: Enterprise Web sites. <http://www1.unece.org/stat/platform/display/bigdata/Report%3A+Enterprise+Web+sites>. Accessed August 21, 2016.

U.S. Census Bureau. (2016). Quarterly Summary of State & Local Taxes. <http://www.census.gov/govs/qtax>. Accessed May 10, 2016.

Ward, G. (2002). Project Gutenberg Moby Word II. <https://github.com/dwyl/english-words>. Accessed May 2, 2016.

Wikipedia. (2016). Unicode. <https://en.wikipedia.org/wiki/Unicode>. Accessed August 21, 2016.

Appendix A: List of State Government Websites

The following is the list of state government websites used to create a universe of PDFs related to government finance, as described in Section 4.1. These websites include home pages of departments of revenue, taxation, and finance and home pages of state comptroller offices. We compiled this list in April 2016. Based on explorations of state and local government websites, other relevant websites may include home pages of state legislatures.

- http://revenue.alabama.gov/
- http://dor.alaska.gov/
- https://www.azdor.gov/
- http://www.dfa.arkansas.gov/Pages/default.aspx
- http://www.sco.ca.gov/
- https://www.colorado.gov/revenue/
- http://www.ct.gov/drs/site/default.asp
- http://revenue.delaware.gov/
- http://otr.cfo.dc.gov/
- http://dor.myflorida.com/Pages/default.aspx
- https://dor.georgia.gov/
- http://tax.hawaii.gov/
- http://tax.idaho.gov/
- http://www.revenue.state.il.us/
- http://www.in.gov/dor/
- https://tax.iowa.gov/
- http://www.ksrevenue.org/
- http://revenue.ky.gov/
- http://revenue.louisiana.gov/
- http://www.maine.gov/revenue/
- http://dat.maryland.gov/Pages/default.aspx
- https://www.mass.gov/dor/
- https://www.michigan.gov/treasury/
- http://www.revenue.state.mn.us/Pages/default.aspx
- http://www.dor.ms.gov/Pages/default.aspx
- http://dor.mo.gov/
- https://revenue.mt.gov/
- http://www.revenue.nebraska.gov/
- http://tax.nv.gov/
- http://revenue.nh.gov/
- http://www.state.nj.us/treasury/
- http://www.tax.newmexico.gov/
- https://www.tax.ny.gov/
- http://www.dornc.com/
- http://www.nd.gov/treasurer/
- http://www.tax.ohio.gov/
- https://www.ok.gov/tax/
- http://www.oregon.gov/dor/pages/index.aspx
- http://www.revenue.pa.gov/Pages/default.aspx
- http://www.dor.ri.gov/
- https://dor.sc.gov/
- http://dor.sd.gov/
- https://www.tn.gov/revenue/
- http://comptroller.texas.gov/
- http://tax.utah.gov/
- http://tax.vermont.gov/
- http://www.tax.virginia.gov/
- http://dor.wa.gov/
- http://www.revenue.wv.gov/Pages/default.aspx
- https://www.revenue.wi.gov/
- http://revenue.wyo.gov/