Journal of Official Statistics, Vol. 37, No. 2, 2021, pp. 395–410, http://dx.doi.org/10.2478/JOS-2021-0017

Applying Machine Learning for Automatic Product Categorization

Andrea Roberson[1]

Every five years, the U.S. Census Bureau conducts the Economic Census, the official count of US businesses and the most extensive collection of data related to business activity. Businesses, policymakers, governments and communities use Economic Census data for economic development, business decisions, and strategic planning. The Economic Census provides key inputs for economic measures such as the Gross Domestic Product and the Producer Price Index. The Economic Census requires businesses to fill out a lengthy questionnaire, including an extended section about the goods and services provided by the business.

To address the challenges of high respondent burden and low survey response rates, we devised a strategy to automatically classify goods and services based on product information provided by the business. We asked several businesses to provide a spreadsheet containing Universal Product Codes and associated text descriptions for the products they sell. We then used natural language processing to classify the products according to the North American Product Classification System. This novel strategy classified text with very high accuracy rates - our best algorithms surpassed over 90%.

Key words: Text analytics; artificial intelligence; data collection.

1. Introduction

The North American Product Classification System (NAPCS) is a comprehensive, hierarchical classification system for products (goods and services) that is consistent across the United States, Canada, and Mexico, and promotes improvements in the identification and classification of products across international classification systems, such as the Central Product Classification System of the United Nations.

Beginning with the 2017 Economic Census, NAPCS will be used to produce economy-wide product tabulations. Respondents were asked to report data from a long, pre-specified list of potential products in a given industry, with some lists containing more than 50 potential products. The definitions of these NAPCS codes can be very complex and ambiguous. Businesses have expressed the desire to alternatively supply Universal Product Codes (UPC) to the U. S. Census Bureau, as they already maintain UPCs in their databases (Thompson and Ellis 2015).

Businesses are generally readily able to report attributes for the business such as total sales, total payroll, and total number of employees. It is much more burdensome for businesses to provide, in a traditional survey instrument, detailed information about their products. Much work has been done around the categorization of products using product

[1] U.S. Census Bureau, 4600 Silver Hill Road, Washington, D.C., 20233, U.S.A. Email: andrea.roberson@census.gov

descriptions (Chen and Warren 2013), but no known study has applied these techniques for the calculation of official statistics (statistics published by government agencies) using only the text of UPC product descriptions. The question we address in this article is: *Can we use UPC codes and their associated descriptions to accurately classify products into NAPCS?*

The main contributions of the article are:

- We present a novel method for survey data collection to automate the U.S. Economic Census with supervised machine learning.
- We provide a product categorization strategy for the three North American countries' classification systems.
- This strategy leverages new approaches and technologies to improve collection speeds, reduce costs, and alleviate respondent burden.

2. Classification Methods

The classification of data requires a class membership decision y' of an unidentified data item x' given some data set $D = (x_1, y_1), \ldots, (x_n, y_n)$ of data elements x_{i} that belong to the class y_i. Here, x_i is a UPC description and $y_i \in \{1, \ldots, P\}$ is its associated NAPCS code. We use three classification algorithms to categorize products: Support Vector Machine (SVM) with linear kernel, Logistic Regression (LR), and Multinomial Naïve Bayes (MNB).

2.1. Support Vector Machines

Support Vector Machines (SVMs) were developed by Vapnik (2000) based on the structural risk minimization principle from statistical learning theory. Statistical learning theory, the backbone of SVMs, provides a new framework for modeling learning algorithms, merges the fields of ML and statistics, and inspires algorithms that overcome many theoretical and computational difficulties. In recent years, SVMs have found a wide range of real-world applications, including face detection from images (Osuna et al. 1997), object recognition (Blanz et al. 1996), speaker identification (Schmidt and Gish 1996; Moreno and Ho 2003), biomedical data classification (Shoker et al. 2005), and text categorization. The many applications of SVMs for text categorization generated considerable research interest in our study.

Joachims (2001) explains how SVMs can achieve good classification performance despite the high-dimensional feature spaces in text classification. The complexity of text-classification tasks are analyzed and sufficient conditions for good generalization performance are identified. The article also provides a formal basis for developing new algorithms that are most appropriate in specific scenarios. The disadvantage of SVMs is that the classification result is purely dichotomous, and no probability of class membership is given (Masood and Al-Jumaily 2013). Another disadvantage of SVMs is the "black box" nature of these classifiers, the decisions made by the model are not easily explainable. The model produced does not naturally provide any useful intuitive reasons about why a particular point is classified in one class rather than another. This leads us to explore models that will be easily understood by our customers. We consider Logistic Regression (LR), where the most important features of the model gives us insights into its inner workings and provides direction for improving performance.

2.2. *Logistic Regression*

Regression modeling is one of several statistical techniques that enable an analyst to predict a response based upon a set of inputs. Linear regression models are commonly used when the range of the response is continuous, and can theoretically take any value. LR, invented in the 19th century for the description of the growth of population and the course of chemical reactions, predicts the probability of an occurrence of an event by fitting data to a logistic curve (Zhang et al. 2011). As the output is restricted to the interval (0, 1), the assumption of an infinite range fails. The logistic function used in this prediction method is useful in that it can take any value from negative infinity to positive infinity as input.

2.3. *Naïve Bayes*

The Naïve Bayes classifier is a classifier based on Bayes Theorem with the naive assumption that features are independent of each other. The Bernoulli Naïve Bayes model uses a set of binary occurrence features. When classifying a text document for example, the Bernoulli Naïve Bayes model is convenient because we could represent the presence or the absence of the given word in the text with a binary feature. On the other hand, this model does not take into account how often the word occurs in the text. The Multinomial Naïve Bayes model (MNB) uses a set of count-based features, each of which does account for how many times a particular feature, such as a word is observed in a document. MNB and SVM are popular choices (Ikonomakis et al. 2005; Joachims 1998; Sebastiani 2002). Both can efficiently deal with high dimensionality and data sparsity.

2.4. *Comparison between Logistic Regression, Naïve Bayes, and Support Vector Machines*

Both Naïve Bayes (Eyheramendy et al. 2003) and Logistic Regression (Zhang et al. 2003) are examples of probabilistic algorithms. Here the dependent variable is a category (Cosmetics or Personal Hygiene). We have a set of text as predictors or features, which come from our UPC product descriptions. This is called training data in ML terminology. MNB takes advantage of probability theory and Bayes' Theorem to predict the NAPCS class. The algorithm is probabilistic, meaning we calculate the probability of each class for a given text, and then output the most likely class. These probabilities are determined by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.

The parameters of an LR model can be estimated by the probabilistic framework called maximum likelihood estimation. Under this framework, a probability distribution for the response variable (NAPCS code) must be assumed, and then a likelihood function defined that calculates the probability of observing the outcome given the input data and the model. This likelihood function can then be optimized to find the set of parameters that gives the greatest possible probability to the training data.

Support Vector Machines, however, are non-probabilistic classifiers. It has the same goal as MNB and LR. Given training data, find the best SVM model, and use the model to classify new UPC descriptions. The difference is that the optimization problem is finding the hyperplane that best separates UPC text labeled "Cosmetics" from those labeled

“Personal Hygiene”. The ML models take the UPC description “*Diamond of California Shelled Pistachios*” and predict the associated NAPCS code for “*snack foods*” as pictured in Figure 1.

3. Relevant Literature

Word (or n-gram) frequencies are typical units of analysis when working with text collections. The general term n-gram means ‘sequence of length n’. A three-word sequence is called a trigram, a sequence of two words is called a bigram, and a single word is called a unigram. It may come as a surprise that reducing a book to a list of word frequencies retains useful information, but this has been demonstrated in natural language processing (NLP) research (Pagliardini et al. 2018). Treating texts as a list of word frequencies (a vector) also makes available a vast range of mathematical tools developed for studying and manipulating vectors.

Text feature extraction is the process of transforming what is essentially a list of words into a feature set that is usable by a classifier. In Bag-of-Words feature selection, the document is treated as an unordered list of words. Under this approach, words are ranked solely by their frequencies. In this case, the set of feature vectors can be considered as a matrix where each row is one instance and each column represents a word found in any of the documents. Thus, each cell (i,j) represents the number of times a word appears in the text of the document. It can be noted that this model builds a $n \times m$ matrix where, for our work, n is the number of UPC text descriptions and m is the number of words without repetition that appear in the n descriptions.

In our analysis, we were able to extract features by using an n-gram model to transform the data into feature vectors for use in our models. We gathered word frequencies (or term frequencies) associated with texts into a document-term matrix using the CountVectorizer class from the scikit-learn python package.

The document-term matrix is usually very high dimensional and sparse. It can create issues for ML algorithms during the learning phase. Therefore, it is recommended to

UPC Description $\Rightarrow$ **NAPCS Code**

DIAMOND OF CALIFORNIA SHELLED PISTACHIOS	snack foods

Which NAPCS code?

Retail sales of candy, prepackaged cookies, and **snack foods**

Retail sales of food **dry goods** and other foods purchased for future consumption (Include flour, sugar, fats and oils, coffee, honey, jams and jellies, pasta, and crackers.)

Retail sales of **fresh fruit and vegetables**

Fig. 1. Example of UPC description Categorization. Given a UPC text description of Pistachios, we seek to predict the associated snack foods NAPCS category. A business might sell pistachios and have trouble deciding what the correct class is: snack foods, dry goods, or fresh fruits and vegetables.

reduce the dimensionality of the data set by either feature selection or dimensionality reduction methods (Wang and Manning 2012). The former selects important features from the original feature set, whereas the latter learns new features from the original set in some other dimension. We will apply Chi-Square (χ^2) and Mutual information (MI) as feature selection methods, and Latent Semantic Analysis (LSA) as a dimensionality reduction technique (Roberts 1996).

3.1. *Chi-Square Dictionary*

The χ^2 test is used in statistics to test the independence of two events. More precisely in feature selection, it is used to test whether the occurrence of a specific term and the occurrence of a specific class are independent. Given a document D we estimate the quantity $\boldsymbol{\chi^2(D, t, c)}$ for each term, and rank them by their score:

$$\chi^2(D, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{1}$$

where e_t takes the value 1 if the document contains term t and 0 otherwise, and e_c takes the value 1 if the document is in class c and 0 otherwise. The χ^2 statistic is a measure of how much expected counts E and observed counts N deviate from each other. A high value of χ^2 indicates that the hypothesis of independence, which implies that the expected and observed counts are similar, is incorrect. Features with very small probabilities deviate significantly from the independence assumption and are therefore considered important. In this context, χ^2 helps identify the most relevant features in the data, where ranking these features may lead to improved classification performance (Bahassine et al. 2018).

3.2. *Mutual Information Dictionary*

In information theory, MI measures how much information a word contains about the class. We might not want to use all the features, but just reliable discriminators (Kozareva 2015). We formally define the MI of a word *w* and a class *c* as *I*(*w*, *c*) where *I* is given by:

$$\sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w p(e_c)} \tag{2}$$

Essentially the MI is a way of capturing the degree of dependence between two variables. MI compares the probability of observing e_w and e_c together with the expected joint distribution if e_w and e_c were independent. MI measures the divergence of the actual joint distribution from the expected distribution under the independence assumption. The larger the divergence is, the higher the MI would be. For each feature we compute the MI, and we repeat this analysis with varying feature sizes.

3.3. *Latent Semantic Analysis Dictionary*

Latent Semantic Analysis (LSA) is a technique for extracting and inferring relations of expected contextual usage of words in documents. LSA takes documents that are semantically similar (talk about the same topics) but are not similar in the vector space, and re-represents them in a reduced vector space in which they have higher similarity. LSA applies Singular Value Decomposition (SVD) to the term-document matrix (TDM). It factors the TDM into three matrices, to form a smaller, low-rank approximation to the original matrix (Bast and Majumdar 2005). The rank-k LSA model of a TDM, $A \in \mathbb{R}^{m \times n}$, is its ran-$k$ SVD,

$$A_k = U_k \Sigma_k V_k^T \tag{3}$$

where $U_k \in \mathbb{R}^{m \times k}$, $\Sigma_k \in \mathbb{R}^{k \times k}$, $V_k \in \mathbb{R}^{n \times k}$, contain the k leading left singular vectors, singular values, and right singular vectors, respectively. In practice, the reduction in size is usually substantial; from a TDM with tens of thousands of documents and terms, to a low-rank approximation with only a few hundred basis vectors for each document (Roberts 1996).

4. Data

To evaluate our approach to test which ML classifier would perform best in product classification, we analyzed 14,000 UPC product descriptions for 44 NAPCS categories, annotated by U.S. Census analysts. The data was provided by a business in the Health and Personal Care Stores sub-sector (class code 446). These data have disproportionate class labels in the response variable. Most of the data is distributed between the cosmetics and personal hygiene NAPCS codes. After removing categories with under 15 products we are left with 16 NAPCS categories. The distribution of our UPC description data is pictured in Figure 2.

Text-based data is inherently unstructured and must be converted to a structured format for predictive modeling or other type of analysis. This was done by applying text processing techniques. Punctuations and numbers were removed from UPC text. Next, all of the letters were converted to lowercase. Another common preprocessing step is the

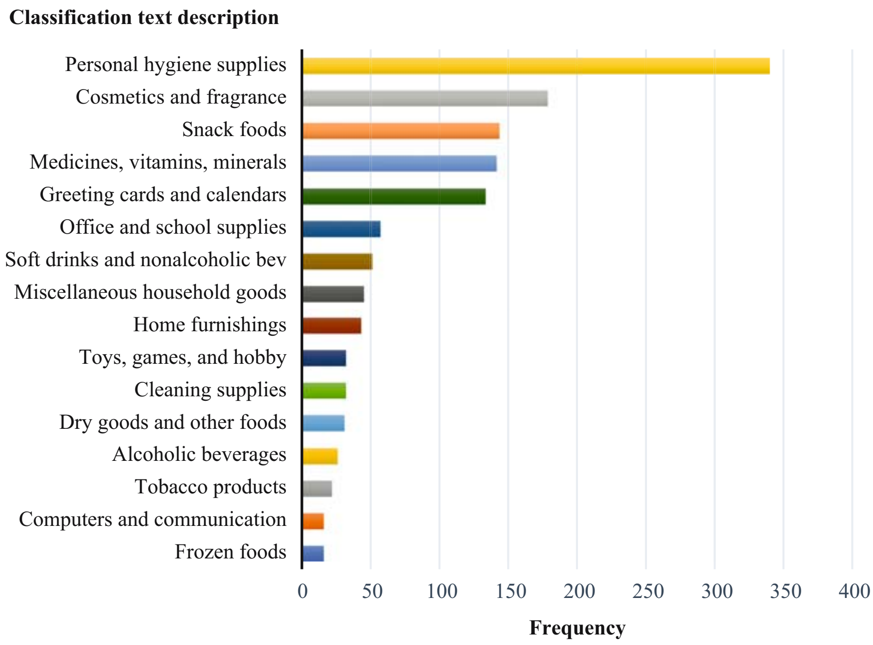

Fig. 2. Product Distribution. This is the distribution of the test data set that was used to measure how well the model generalizes to unseen data.

removal of white space. It is typically the result of all the left over spaces or tabs that were not removed along with the words that were deleted. All white space was removed.

A further preprocessing technique is the removal of stop words. They are words which are filtered out before or after processing of natural language data (text). Any group of words can be chosen as the stop words for a given purpose. Stop words are words that are so common in a language that their information value is almost zero, that is, they do not carry significant information. We would not want these words taking up space in our database, or taking up valuable processing time. Some examples are “a”, “about”, “be”, “do”. Therefore, it is recommended to remove them before further analysis.

5. Implementing Machine Learning Algorithms

The most widely used library for implementing ML algorithms in Python is scikit-learn. This library is a Python module integrating a wide range of state-of-the-art ML. This package focuses on bringing ML to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency.

A well-fitted model should not just provide good prediction accuracy on the data it was fitted to, it should also generalize to data not yet seen. We can estimate this generalization accuracy with a technique called cross-validation. The simplest form of cross-validation is as follows: the data are separated into a training set and a test set. The algorithm is fit on the training set and the accuracy (e.g., the percent is correctly classified) is evaluated on the test set, giving an estimate of how the fit generalizes.

All classifiers have various parameters which can be tuned to obtain optimal performance. Tuning is performed for varying values of the tuning parameters, searching for those that give the best generalization accuracy (Guenther and Schonlau 2016). This can be done by choosing a small number of possible values to test for each parameter, and trying all possibilities on the grid of their combinations. This is known as a grid search. In the context of ML, hyperparameters are parameters whose values are set prior to the commencement of the learning process. In scikit-learn, hyperparameter tuning can be conveniently done with the GridSearchCV estimator. It takes as input an estimator (such as accuracy) and a set of candidate hyperparameters. Cross-validation scores are then computed for all hyperparameter combinations, in order to find the best one. In this research, we tune the LR, MNB, and SVMs with GridSearchCV.

For LR, we use the sklearn.linear_model.LogisticRegression package in the scikit-learn library.

The parameters are as follows:

- penalty: It specifies the norm used in penalization. It can be ‘l1’, or ‘l2‘. The default value is ‘l2’.
- C: It is the inverse of the regularization strength. Smaller values specify stronger regularization.

We first observe that setting the parameter C is crucial as performance drops for inappropriate values of C. The LR regularization parameter was set in the range of ($C = 10^{-4}, 10^{-3}, \ldots, 10^{5}, 10^{6}$). A large C can lead to an overfit model, while a small C can lead to an underfit model. We used GridSearchCV with 10-fold cross-validation to tune C in this hyperparameter space.

The package used for SVM classification in the scikit-learn library is svm.SVC.

The parameters are as follows:

- C: It is the regularization parameter, C, of the error term.
- kernel: It specifies the kernel type to be used in the algorithm. It can be 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed', or callable. The default value is 'rbf'.
- degree: It is the degree of the polynomial kernel function ('poly') and is ignored by all other kernels. The default value is 3.
- gamma: It is the kernel coefficient for 'rbf, 'poly', and 'sigmoid'. If *gamma* is 'auto', then $\frac{1}{\mathrm{n}}$ features will be used instead.

Training SVMs with a linear kernel is faster than with any other kernel. When you train an SVM with a linear kernel, you only need to optimize the C regularization parameter. When training with other kernels, you also need to optimize the *gamma* parameter, which means that peforming a grid search will usually take more time. Therefore, linear kernels are indeed very well suited for text-categorization. It should be kept in mind, however, that this is not the only solution. In some cases, using another kernel might be better. The recommended approach for text classification is to try a linear kernel first, because of its advantages. An SVM with a linear kernel is similar to logistic regression. Therefore, in practice, the benefit of SVMs typically comes from using non-linear kernels to model non-linear decision boundaries. In this study, in an effort to get the best possible classification performance, it was of interest to try the other kernels to see if accuracy was improved.

We did a set of experiments with different kernel functions such as the linear, RBF, polynomial, and sigmoid in order to see the quality of generalization for each kernel function. Using sklearn's SVM implementation svm.SVC, we apply a grid-search to find the best pair (*C, gamma*) for each kernel function using 10-fold cross-validation. In order to increase efficiency, we try exponentially growing sequences of (*C, gamma*) to identify good parameters ($C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$; $gamma = 2^{-15}, 2^{-12}, \ldots, 2^{12}$). After the optimal (*C, gamma*) is found, the training data is trained using the SVMs with different kernels and the best parameters to generate the final models. After testing our SVM algorithm with various kernel transformations, we identified the linear kernel as the most efficient kernel that resulted in the highest classification results.

For MNB, we use the sklearn.naive_bayes.MultinomialNB package in this scikit-learn library.

The parameters are as follows:

- alpha: It specifies the smoothing parameter. The default value is '0'.

The alpha parameter controls the level of smoothing applied in the training set. This can be useful when items in the test set would have zero probability based on the training set. There is no good rule of thumb for setting this parameter, so the experiment included several values within a parameter grid search.

6. Experiments and Results Data

Feature selection is simply selecting and excluding given features without changing them. Dimensionality reduction transforms features into a lower dimension. Table 1 shows a

Table 1. Summary of feature selection and classification methods.

Feature reduction methods	Abbreviation	Classifiers	Abbreviation
Chi-square feature selection	χ^2	Support vector machine	SVM
	χ^2	Logistic regression	LR
	χ^2	Multinomial naïve bayes	MNB
Mutual information feature selection	MI	Support vector machine	SVM
	MI	Logistic regression	LR
	MI	Multinomial naïve bayes	MNB
Latent semantic analysis dimensionality reduction	LSA	Support vector machine	SVM
	LSA	Logistic regression	LR

summary of the feature selection and dimensionality reduction techniques that are considered for each classifier.

6.1. Results

We define classification accuracy as the percentage of UPC codes for which the classification agreed with the known categories. A UPC code description, whose fitted state differs from the ground truth label, is defined as an error. Classification accuracy, precision, recall, and F-scores were used as performance metrics. Precision captures the fraction of UPC codes classified as cosmetics that are truly cosmetics. Recall captures the fraction of codes that are truly cosmetic, that are found by the model. The F-score is the weighted harmonic mean of precision and recall.

We also calculated Cohen's Kappa coefficient to measure agreement, beyond chance, between the fitted results and the ground truth data (Grandini et al. 2020). Kappa values range between − 1 (all text incorrectly classified) and 1 (all text correctly classified). A Kappa value equal to zero indicates a performance no better than random. This was especially important since our data set has severe class imbalance, a classifier could obtain high accuracy by always guessing the most frequent class. Table 2 shows the accuracy rates obtained on various feature sets, across all considered classification and feature selection models. For the LSA dimensionally reduced data, it no longer makes sense to use MNB, since the features are no longer valued in positive integers. However, we can still use SVM and LR for classification (Tong and Koller 2002).

Performance at feature set size 1,100 showed that all models received a boost of 5 percentage points or more using the LSA model, with SVM and LR tied for best. The frequency based Unigram model, Chi-Square, and MI model achieved very good results with a feature set of 1,100, but LSA demonstrated much better results than any other model.

LR, SVM, and MNB displayed a large increase in accuracy at the feature set size of 4,100. With the exception of the bigram model, and the combined unigram and bigram model, all our methods surpassed 90% accuracy with the feature set of 4,100. The optimal feature set size, however, seems to be at 9,100, the entire vocabulary, where we continue to see growth in accuracy without any overfitting. SVM was the overall best performer as the feature set size increased, marginally beating LR and MNB by at most two percentage points. Table 3 summarizes performance metrics for the frequency based unigram model. Precision, Recall, and the F-scores are also all above 90%. All of the Kappa values indicate

Table 2. Accuracy Comparison. We can observe that generally, SVM and LR perform Better than MNB.

Classifier	100 words	1,100 words	4,100 words	7,100 words	9,100 words
Chi-square feature selection with unigrams					
SVM	.55	.83	.94	.96	.96
LR	.55	.83	.93	.95	.95
MNB	.49	.79	.92	.94	.94
Mutual information feature selection with unigrams					
SVM	.64	.87	.95	.95	.96
LR	.64	.87	.94	.94	.95
MNB	.57	.83	.93	.94	.94
LSA dimensionality reduction with unigrams					
SVM	76	.92	.95	.96	.96
LR	.77	.92	.95	.95	.95
Unigrams					
SVM	.62	.87	.95	.95	.96
LR	.62	.86	.94	.95	.95
MNB	.60	.83	.93	.93	.93
Bigrams					
SVM	.47	.62	.71	75	76
LR	.47	.62	.70	75	76
MNB	.47	.62	.71	75	77
Unigrams+bigrams					
SVM	.61	.84	.93	.95	.95
LR	.61	.85	.93	.94	.95
MNB	.60	.83	.92	.93	.93

Table 3. Classification results using 9,100 features. SVM slightly outperforms the other classifiers.

Classifier	Precision	Recall	F-score	Kappa
Unigrams				
SVM	.96	.96	.96	.94
LR	.95	.95	.95	.94
MNB	.93	.93	.93	.92

that the model results were not due to chance. While dimension reduction wasn't useful for this data set, this serves as an important benchmark for future data collection.

6.2. *Error Analysis*

Examples closer to the decision boundary are frequently misclassified, that is they are more difficult to identify. Figure 3 shows some misclassified examples. The UPC Description *No7 Lip Balm Coral Blossom* was predicted to belong to the personal hygiene NAPCS code. The lip balm pictured in Figure 3 closely resembles the packaging of lipstick. It may be useful in the future to consider the brand of a company like *No7* (a cosmetics brand), to more accurately predict NAPCS.

We also noticed errors due to annotator disagreement, subject experts determined *Colorsilk H/C Light Blonde* was mislabeled as Cosmetics. Respondent and annotator inconsistencies will be eliminated by creating an official gold standard.

To validate our model further, we looked at which features it is using to make decisions. Figure 4 shows the most important abbreviations in the UPC text data for the cosmetics NAPCS. Cover Girl (cg), Maybelline (mayb), Revlon (rev), are major cosmetics brands. This demonstrates our model is making understandable decisions.

UPC Description	Actual	Predicted
NEUT MEN SKN CLRNG ACNE WSH 5.1OZ	5001425000	5001450000
CALDESENE PROTECTING POWDER 5OZ	5001425000	5001450000
COLORSILK BTTRCRM NAT BRWN 60/51N	5001425000	5001450000
NO7 LIP BALM CORAL BLOSSOM .09OZ	5001450000	5001425000
DR FRED SUMMIT SKN/WHITNR 202 2OZ	5001450000	5001425000
COLORSILK H/C LIGHT BLONDE #81	5001450000	5001425000

Fig. 3. Misclassified Examples. Most of the model's mistakes were between the cosmetics NAPCS code (5001450000) and the personal hygiene code (5001425000). Personal hygiene was predicted as cosmetics nine times. Cosmetics was predicted as personal hygiene five times.

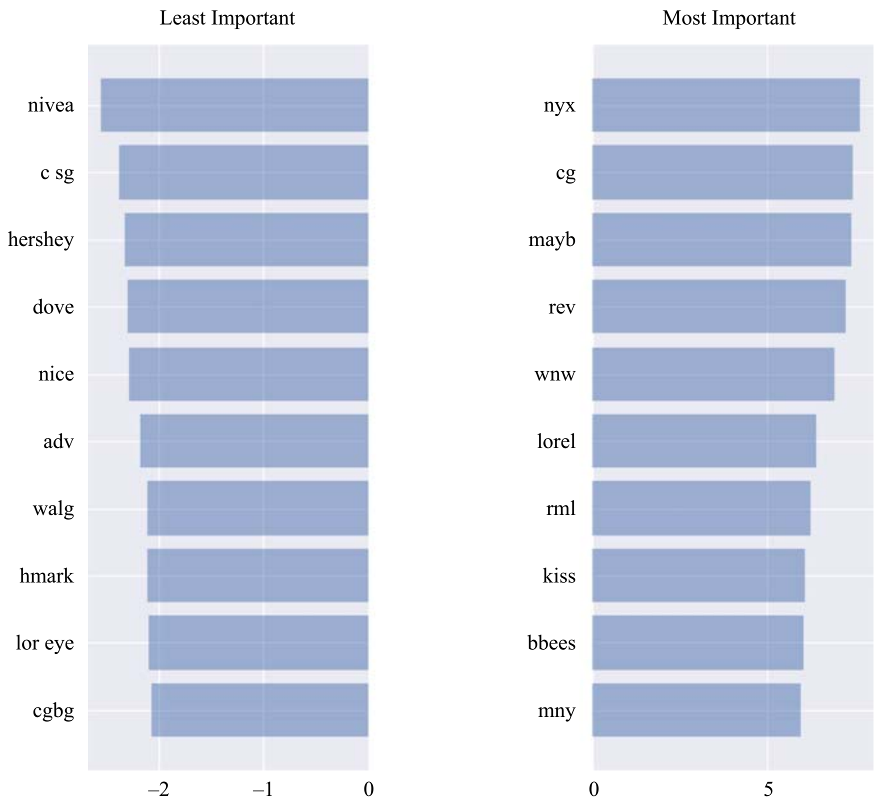

Fig. 4. Importance Scores for Cosmetics NAPCS. On the left we have the ten least important features, the ten most important are shown on the right. The most important feature was nyx, the least important was nivea.

7. Conclusions

The economic census – based on a representative sample of approximately four million businesses from about 400 industries – is vital to understanding how the nonagricultural sectors of the US economy are performing. Statistics from the economic census are used by policymakers and trade and business associations, as well as individual business owners. In 2017 theeconomic census was updated to collect data on more than 8,000 products through the NAPCS classification system. Due to the nature of this survey change, which included detailed product data collection, response rates were lower than expected.

The challenge of collecting the same product data in efficient, innovative, and less costly ways motivated this study. Respondents have found it burdensome to report data from a long, prespecified list of potential products. To this end, we turn to modern methods and techniques for data collection and classification. We consider integrating alternate data sources for survey collection processes, and an automated survey questionnaire system.

We examine three commonly used ML models for text classification; LR, MNB, and SVM in order to predict NAPCS codes. We found the best performing model is SVM. All of the experiments were carried out on a 2.7GHz Intel 4-Core i7-6820 CPU with 16 GB of RAM, using scikit-learn software and Microsoft Windows 10 operating system. Very high processing speeds were achieved in the learning phase. The processing time was under 180 ms on a 2.7 GHz PC. The best performing SVM model achieves a good accuracy above 95%, but there is room for improvement.

The year 2018 was an inflection point for truly groundbreaking new developments for deep learning approaches to NLP. Traditionally, NLP models were trained after random initialization of the model parameters (also called weights). Transfer learning is a concept in deep learning where you take knowledge gained from one problem and apply it to a similar problem. A technique where instead of training a model from scratch, we use models pre-trained on a large data set and then fine-tune them for specific NLP tasks (Malte and Ratadiya 2019).

The Google AI paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding collected honors from the ML community. Google researchers (Devlin et al. 2019) present a deep bidirectional Transformer model that redefines the state-of-the-art for eleven NLP tasks, surpassing human performance in the challenging area of question answering. Language models learn the sequential nature of language, the nuanced flow of words and sentences, and the context in which words appear (Malte and Ratadiya 2019). These models have a much more nuanced understanding of language than prior approaches.

BERT (Devlin et al. 2019), ULMFiT (Howard and Ruder 2018), and GPT (Radford et al. 2018) have shown excellent results on multiple NLP tasks like sentiment analysis, sentence similarity, and document multi-label classification. Relying on deep neural networks, various research groups trained these new models with tremendous amounts of data and compute power.

The hope is that transfer learning for NLP will significantly reduce the effort to create systems dealing with text classification. Unfortunately, in practice, it is not so trivial. BERT is a huge model with more than 100 million parameters. We not only need a GPU to fine tune it, but also at inference time, a CPU is not enough. Using large models in a production environment requiring large amounts of compute and memory, is currently impractical for our National Statistics Office (Zafrir et al. 2019). In future work, we hope to explore these models.

LR remains the clear choice when the primary goal of model development is to look for possible causal relationships between independent and dependent variables, and a modeler wishes to easily understand the effect of predictor variables on the outcome given that the model equation is also provided. This will be useful to both analysts reviewing our predictions, and also future performance assessments.

We believe we can achieve better consistency, and response by automating classification for US businesses. We think this is a more precise strategy as indicated by our high accuracy rates. This research yields promising results and potentially reduces administration cost in survey processing. We have demonstrated a more efficient methodology for classifying NAPCS that can also be considered by other North American countries, and other official statistics agencies. This work also lays the foundation for not collecting this information only every five years, but with the timeliness that intelligent predictions provide.

8. Future Work

Our next steps include obtaining UPC data from other businesses such as a clothing and electronics store. We plan to test the models on additional data sets. If these methods continue to perform well, we will consider system-to-system agreements with several companies. Our vision is to have more companies opting into the system-to-system arrangement over time.

While there are thousands of papers published each year on how to design and train models, there is surprisingly less research on how to manage and deploy such models in production. The result of model development is typically trained models that can be used to render predictions as part of some application or service (Lee et al. 2018). The final phase of rendering predictions is often referred to as prediction serving, model scoring, or inference. Prediction serving requires integrating ML software with other systems including user-facing application code, live databases, and high-volume data streams. Due to domain-specific requirements, prediction serving has been widely studied in areas such as ad-targeting and content-recommendation (Agarwal et al. 2015). These targeted approaches do not address the full set of system challenges necessary to developing high value ML applications. We require specialized solutions engineered for our unique customer needs.

Our future work includes providing practical lessons for developing ML applications, whether you are developing your own prediction serving system or using off-the-shelf software. We also seek to incorporate the Census general-purpose low-latency prediction serving system for automatic product classification. Census currently uses an application programing interface (API), in conjunction with a web-based survey instrument, to apply ML with Logistic Regression to survey responses in real-time. The current system in production has achieved low latency, high throughput and graceful performance degradation under heavy load. Looking toward the 2022 Economic Census, we want to reduce respondent burden while providing the data of greatest need to stakeholders.

9. References

Agarwal, D., B. Long, J. Traupman, D. Xin, and L. Zhang. 2015. "LASER: A Scalable Response Prediction Platform for Online Advertising." In Proceedings of the 7th ACM International Conference on Web Search and Data Mining. February 2014, New York, NY, USA: 173–182. DOI: https://doi.org/10.1145/2556195.2556252.

Bahassine, S., A. Madani, M. Al-Sarem, and M. Kissi. 2018. "Feature selection using an improved Chi-square for Arabic text classification." *Journal of King Saud University-Computer and Information Sciences*: 1319–1578. DOI: https://doi.org/10.1016/j.jksuci.2018.05.010.

Bast, H., and D. Majumdar. 2005. "Why spectral retrieval works." In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 2005, Salvador, Brazil: 11–18. DOI: http://doi.acm.org.ezproxy.lib.unimelb.edu.au/10.1145/1076034.1076040.

Blanz, V., B. Scholokopf, H. Bulthoff, C. Burges, V.N. Vapnik, and T. Vetter. 1996. "Comparison of view-based object recognition algorithms using realistic 3D models." In Proceedings of International Conference on Artificial Neural Networks-ICNN'96, July 1996, Berlin, Germany: 251–256. DOI: https://doi.org/10.1007/3-540-61510-5_45.

Chen, J., and D. Warren. 2013. "Cost-sensitive learning for large-scale hierarchical classification of commercial products," In Proceedings of 22nd Conference on Information and Knowledge Management (CIKM2013) 1351–1360. DOI: https://doi.org/10.1145/2505515.2505582.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. 2019. "Bert: Pretraining of deep bidirectional transformers for language understanding." In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, June 2019, Minneapolis, USA: 4171–4186. DOI: https://www.aclweb.org/anthology/N19-1423/.

Eyheramendy, S., D. Lewis, and D. Madigan. 2003. "On the Naïve Bayes model for text categorization." In Proceedings of the 9th Workshop on Artificial Intelligence, January 2003, Key West, USA: 93–100. Available at: http://proceedings.mlr.press/r4/eyheramendy03a/eyheramendy03a.pdf.

Grandini M., E. Bagli, and G. Visani. 2020. "Metrics for Multi-Class Classification: an Overview." Available at: arXiv preprint arXiv:2008.05756.

Guenther, N., and M. Schonlau. 2016. "Support vector machines." *Stata Journal* 16: 917–937. DOI: https://doi.org/10.1177/1536867X1601600407.

Howard, J., and S. Ruder. 2018. "Universal Language Model Fine-tuning for Text Classification." In Proceedings of ACL.

Ikonomakis, M., S. Kotsiantis, and V. Tampakas. 2005. "Text classification using machine learning techniques." *WSEAS Transactions on Computers* 8(4): 966–974. DOI: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9153&rep=rep1&type=pdf.

Joachims, T. 1998. "Text categorization with Support Vector Machines: Learning with many relevant features." In: *Machine Learning: Lecture Notes in Computer Science*. vol 1398. Springer: Berlin and Heidelberg.

Joachims, T. 2001. "A statistical Learning model of text classification for Support Vector Machines." In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: SIGIR 2001, September 9–13, 2001, New Orleans, LA, USA: 128–136.

Kozareva, Z. 2015. “Everyone likes shopping! Multi-class product categorization for e-commerce.” In Proceedings of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL: 1329–1333. DOI: https://doi.org/10.3115/v1/N15-1147.

Lee, Y., A. Scholari, B. Chun, D. Santambrogio, M. Weimer, and M. Interlandi. 2018. “PRETZEL: Opening the black box of Machine Learning Prediction Serving.” *13th USENIX Symposium on Operating Systems Design and Implementation*. October 2018, Berkeley, USA: 611–626. DOI: https://dl.acm.org/doi/10.5555/3291168.3291213.

Malte, A., and P. Ratadiya. 2019. “Evolution of transfer learning in natural language processing.” Available at: arXiv preprint arXiv:1910.07370 [cs.CL] Accessed 20 August 2019.

Masood, A., and A. Al-Jumaily. 2013. “Computer aided diagnostic support system for skin cancer: a review of techniques and algorithms.” *International Journal of Biomedical Imaging*: 1–22. DOI: https://www.hindawi.com/journals/ijbi/2013/323268/.

Moreno, P., and P. Ho. 2003. “A New SVM Approach to Speaker Identification and Verification Using Probabilistic Distance Kernels.” In *Eurospeech*: 2965–2968. DOI: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.2378&rep=rep1&type=pdf.

Osuna, E., R. Freund, and F. Girosi. 1997. “Support Vector Machines: Training and Applications”, A.I. Memo No. 1602, Artificial Intelligence Laboratory, MIT. DOI: https://ieeexplore.ieee.org/document/622408.

Pagliardini, M., P. Gupta, and M. Jaggi. 2018. “Unsupervised learning of sentence embeddings using compositional n-gram features.” Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, June 2018, New Orleans, USA : 528–540. DOI: https://www.aclweb.org/anthology/N18-1049.

Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever. 2018. “Improving language understanding by generative pre-training.” In The Handbook of Contemporary Semantic Theory. editor: S. Lappin. Blackwell, Cambridge MA & Oxford.

Roberts, C. 1996. “Anaphora in Intensional Contexts.” In *The Handbook of Contemporary Semantic Theory*, edited by S. Lappin. Cambridge MA & Oxford: Blackwell.

Schmidt, M., and H. Gish. 1996. “Speaker identification via support vector classifiers.” In IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP, May 1996, Washington, USA: 105–108. DOI: https://ieeexplore.ieee.org/document/540301.

Sebastiani, F. 2002. “Machine Learning in Automated Text Categorization.” *ACM Computing Surveys* 34: 1–47. DOI: https://doi.org/10.1145/505282.505283.

Shoker, L., S. Sanei, and J. Chambers. 2005. “Artifact removal from electroencephalograms using a hybrid BSS-SVM algorithm.” In *IEEE Signal Processing Letters*: 721–724. DOI: https://doi.org/10.1109/LSP.2005.855539.

Thompson, K., and Y. Ellis. 2015. Exploratory Data Analysis of Economic Census Products: Methods and Results. In JSM Proceedings, Survey Research Methods

Section, American Statistical Association, Seattle, WA, August 7–13. Alexandria, USA: 828–842. DOI: http://www.asasrms.org/Proceedings/y2015/files/233942.pdf.

Tong, S., and D. Koller. 2002. “Support vector machine active learning with applications to text classification.” In *The Journal of Machine Learning Research (Volume 2)*: 45–46. DOI: https://doi.org/10.1162/153244302760185243.

Vapnik, V.N. 2000. *The Nature of Statistical Learning Theory*, (2nd edition). New York: Springer.

Wang, S., and C.D. Manning. 2012. “Baselines and bigrams. Simple, good sentiment and topic classification.” In Proceedings of the 50th Annual Meeting of the Association for Computatioral Linguistics (Volume 2: Short Papers), Association for Computational Linguistics: 90–94.

Zafrir, O., G. Boudoukh, P. Izsak, and M. Wasserblat. 2019. “Q8BERT. Quantized 8Bit BERT.” Available at: arXiv:191O.O6188 [cs.CL]. Accessed 30 November 2019.

Zhang, J., R. Jin, Y. Yang, and A.G. Hauptmann. 2003. “Modified Logistic Regression. An Approximation to SVM and Its Applications in Large-Scale Text Categorization,” In Proceedings of the 20th International Conference on Machine Learning. August 21–24, Washington D.C., USA: 888–895. Available at: arXiv:1910.06188 [cs.CL] (accessed 30 November 2019).

Zhang, M., G. Johnson, and J. Wang. 2011. “Predicting Takeover Success Using ML Techniques.” Journal of Business & Economics Research (JBER).

Received May 2019
Revised November 2019
Accepted March 2020