Understanding Open Ended Survey Data using Machine Learning Techniques

Lisa Wang, Isaac Caswell, Avi Bagla
Stanford University
{lisa1010, icaswell, abagla}@stanford.edu

Abstract

The authors address the problem of automatically assigning codes to open ended responses to political surveys using machine learning approaches. The data consist of ten datasets of about 1000-2000 transcripts each, corresponding to different survey questions, with up to 80 possible codes each. A variety of classifiers, features and data augmentations are explored. The most effective technique is found to be (oversampling) logistic regression on unigrams, bigrams, wordNet synonyms, sentiment and spellcheck features, which achieves an F1 of .57-.98, depending on the dataset. This wide range highlights the distinct character of each dataset.

1 Introduction and Background

In the field of political science, knowledge about events and political leanings are learned from prose. Researchers will conduct surveys or comb through speeches conducted by politicians to gain insight into the views of large populations of people and the current events of the time. Historically, this is done by humans. People are hired to read text and categorize it into predetermined categories in a process that social scientists refer to as "coding". Because of the sheer size of these corpora and the care that must go into each decision, coding is inefficient and expensive. As a result, there has been increased interest in applying machine learning (ML) and natural language processing (NLP) techniques to these problems.

Much of the conversation around machine learning in political science is a debate about its effectiveness. The potential is certainly there: Because of its wide commercial applicability in industries including search engines, marketing, spam flagging and many others, the field of text classification (TC), usually in the form of subjectivity analysis (opinion mining, sentiment analysis), has recently boomed as a field (Tsytsarau & Palpanas, 2012). As blogs, comments and other user-generated content are so widespread, TC has been used as a tool to capture opinions at a larger scale, which has made it a potential candidate for classifying political prose. However, one of the biggest challenges is translating one classification application to a different domain (Glorot et al, 2011). For these reasons, many political scientists are skeptical of the promise of machine learning - as Grimmer and Stewart (2013) put forth. Many of these concerns arise from the assumptions made by language models, and from the subtlety of political language.

In this paper we present several machine learning approaches which are not particular to any one domain, outside of the need for domain-specific training data. Specifically, we present a pipeline of **preprocessing** $\rightarrow$ **data augmentation** $\rightarrow$ **featurization** $\rightarrow$ **training**, which we apply equally to thirteen distinct datasets, without any adaptation to any specific one. This can be seen as a general framework upon which to build more specific models, but is also effective in its own right. For instance, for all datasets about political knowledge of a public figure, this system achieves over .90 F1—but for the survey asking about respondents' occupation, it peaks at .57 F1.

2 Problem statement

Our objective is to automatically label transcripts of open ended survey data with codes corresponding to their political meaning. We look specifically at ANES (American National Election Studies) data, where people were surveyed about their opinions on a variety of political issues.

3 Previous Approaches

The problem of working with open-ended survey data has already been approached by some researchers. A simple technique proposed in 2013 was phrase matching to custom made keyphrases, as seen in DeBell's paper. In other parts of the literature machine learning aproaches tend to perform the best, with SVM and decision trees leading the way. Structural Topic Models (an extension of the well-known LDA) (Roberts et al, 2014) have also been used to moderate success. King and Lowe (2003) attempt something similar, using IDEA categorization to categorize topics. A variety of other techniques stem from these two, but ultimately nothing seems to significantly outdo SVM.(King & Lowe, 2003)

4 Overview of Data Used

4.1 Data Sets

Our data include thirteen sets of coded interviewer transcripts and coding instructions from the 2008 ANES Study. Each set contains approximately 2000 transcripts to a single open-ended survey question and the codes that a human coder assigned to them. The complete list of thirteen includes questions on current and past occupation and industry of the respondent, the most important political and personal problem the respondant sees in the nation, political knowledge questions about political figures (e.g. *"The first name is DICK CHENEY. What job or political office does he NOW hold?"*), and a question about the motivation behind the terrorist attack on September 11th, 2001. In order to make this more manageable, we will restrict most of our analysis to the following four datasets:

1. **occupation:** A dataset which combines questions as to current and previous occupation of the respondant. The question asked is *"What kind of work did you do on your [last/current] regular job? What was your occupation? What were your most important activities or duties?"*
2. **mip_political:** A dataset which combines answers to the questions *"What do you think is the most important political problem facing the United States today?"* and *"What do you think is the second most important political problem facing the United States today?"*
3. **pk_cheney:** Answers to the question *"The first name is DICK CHENEY. What job or political office does he NOW hold?"*
4. **terrorists:** Answers to the question *"As you know, on September 11th 2001, a group of Terrorists took control of several U.S. commercial airplanes and crashed them into the World Trade Center in New York and the Pentagon in Washington. What do you think the Terrorists were trying to accomplish by their actions?"*

We chose these four data sets to cover different types of questions and machine learning challenges that appeared in our data. The transcripts in the data set "occupation" tend to be short (e.g. *"bartender"*) or very specific (e.g. *"production control data entry"*), and there are 99 possible codes. As a result, this is the hardest dataset to work with, and gave us the poorest results of any dataset. On the other hand, "PK Cheney" is the dataset we do the best on, likely because the results are very predictable: 967 out of 2095 responses contained the string "vice pres". "mip [Most Important Problem] political" is very similar to "mip personal", in fact they have the same coding instructions and the same set of codes. We arbitrarily chose one of them to represent this type of question. The "mip" questions are among the most open-ended questions and we predicted that these would be harder to classify than e.g. the transcripts in "occupation". "Terrorists" did not fit into any of the three previously mentioned question types, so we included it in our model data sets as well.

4.2 Coding Instructions

For every open-ended question, we also received the corresponding coding instructions which specify how to code the transcripts. The coding instructions include tables with examples for each available code. Table 1 is an excerpt from the coding instructions.

4.3 Peculiarities of the Data

An important thing to note about these data sets is that responses tend to be short and messy. To illustrate the first, Figure 1 demonstrates an approximate power law distribution over lengths of transcripts in the mip_political dataset, with a whopping 1259 length-1 transcripts, tapering down quickly to only 72 length five transcripts, dropping into the single digits already after nine words. As an illustration of the messiness we present a brief list of ways that 'economy' was spelled in

ID	Code	Examples
4	Kill people	Kill, Kill people
5	Prove that the United States is weak	US is weak, US is vulnerable, US not strong, US can be defeated, US can be hurt, the terrorists are strong, terrorists are not weak, terrorists are big, terrorists can hurt the US
6	Start a war with the United States	Start a war with the US, Get the US involved in a war

Table 1: Coding instructions excerpt for the data set "terrorists"

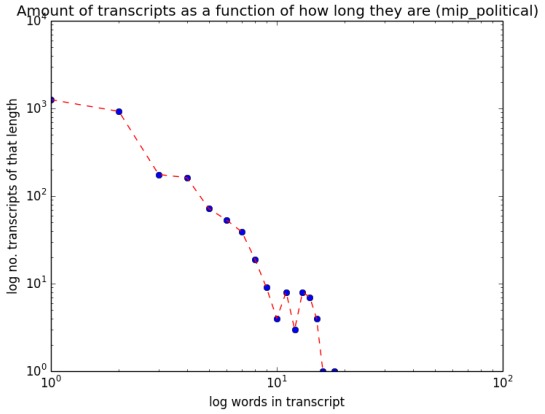

Figure 1: Length of transcripts follows an approximate power law. (data from 2100 transcripts)

the mip_political dataset: **[Economy1, econami, the eocomic, econemy, econmy, the econamic, the econoicy, the econminc, the econamiy,the econoacia, econacia, the economic, The eocnomy, the e conomy, econoy]**. An important thing to note is that many of the misspellings are very far off the intended word (Levenshtein distance of greater than two), and because of the telegraphic nature of the data, there is very little distributional information to work with.

5 Methods

After converting the ANES data into machine readable text, we looked into methods of classification and feature extraction.

5.1 Preprocessing

Most of the preprocessing was to get the data from unstandardized word documents into a machine-readable format. Following that we lowercased the data and split it on a set of punctuation we collected from the training data, including commas, semicolons and double slashes. We experimented with stemming, and NER tagging, but this sewmed to hurt our performance; it is worth further inquiry as to why this was so. (We applied spellchecking at the featurization level, but it could just as well have been done here.)

5.2 Data augmentation

Since each data set only contains about 2000 transcripts, we came up with three strategies to augment our data.

1. *Creating additional transcripts from the examples in the coding instructions.* The coding instructions contain examples for each code. To make use of these instructions, we turned each example into a new transcript with its corresponding code as the gold code, and added it to the dataset.

2. *Combining data sets with identical coding instructions.* Some data sets such as most important political problem and most important personal problem had the same set of codes, so we combined the two data sets. We took a similar approach for current and past occupation as well as current and past industry.

3. *Oversampling* The data are very unbalanced, meaning that certain codes only have one or two transcripts assigned to them, whereas the most common one (such as 'economy' for mip_political) might have over a thousand. Oversampling is a method to deal with unbalanced datasets, and involves making duplicates of examples from the underrepresented classes, and training on the augmented dataset. We oversample at a rate of 6x using a streaming algorithm we developed, which compares the current distribution of classes at any point to the uniform distribution, and compensates accordingly.

5.3 Classifiers

Our models contrast a keyphrase-overlap style model, similar to that proposed by DeBell, with standard machine learning approaches. For each machine learning classifier, we ran grid search over a range of parameters using 8-fold cross validation, and in all tests thenceforward used those

parameter values. Following is a description of the classifier we used, along with their parameters. The baseline was implemented by hand, and all machine learning approaches are implementations from the scikit-learn library.

1. **Keyphrase Baseline** During 'training', the baseline stores a mapping of code name to a set of words related to each code. There are two mappings for each code: one is the description of the code, including the name of the code, and the other is a short list of examples given in the instructions. To classify a transcript, we first compute the Tanimoto distance (Jaccard similarity) between the set of unigrams in the transcript and the description of each code, and also between the examples from each code. The code assigned corresponds to whichever code yields the greatest overlap with the transcript.

   The Tanimoto distance is defined as the size of the intersection of two sets divided by the size of their union. It is essentially a measure of how many words two sets have in common, scaled by how long the sets are, to avoid giving the advantage to longer descriptions.

2. **Logistic Regression**

   Logistic regression is a common classifier in text classification, which learns a hyperplane separate different classes of examples. In two dimensions, one can think of it as finding a straight line which separates different clusters of points.

   Results of grid search indicated that L1 penalty with a one-versus-rest multiclass scheme (using cross-entropy loss) was the most effective (with lbfgs solver). The optimal value of the parameter C, the inverse regularization strength, varied from dataset to dataset from as low as .6 to as high as 3.7. For our tests, we set it at 3.7.

3. **Support Vector Machines** Support Vector Machines (SVM) are one of the most popular out of the box machine learning techniques on the market, and although they tend to train more slowly than analogous models, they are especially good at dealing with noisy data.

   Grid search lead us from very mediocre performance to quite good performance. The final model used a linear kernel and a value of C=1.0, and used the shrinking heuristic.

4. **K Nearest Neighbors** K nearest neighbors is a very simple classifier, although in certain domains is hard to beat. It stores all the data it saw during training, and during the testing phase, classifies a new data point based on the classes of the nearest points from the training data. The most important hyperparameter for KNN is the distance metric, which defines what it means for two points to be close. Our best models used the Minkowski distance metric (thought Manhattan beat it in some cases). Across all datasets, the best value for K, the number of neighbors to consider, was 1, suggesting that the domain might be characterized by a high multiplicity of identical responses, or very similar responses with different codes.

5. **Random Forest** A decision tree decides what class a thing is by sequentially looking at different attributes; a random forest takes predictions from an ensemble of decision trees, and averages the result, thereby reducing overfitting. Our parameters after grid search were to use 50 estimators, splitting on the gini criterion without bootstrapping, with the maximum features to consider when making a split being sqrt(n).

6. **Adaboost classifier** The Adaboost classifier is in fact a meta algorithm which trains a ensemble of classifiers, adjusting them depending on what they get wrong. There is no doubt an art to applying it well; its performance was so poor in our case that we dropped it early.

5.4 Features

We experimented with a variety of different feature functions. For most datasets, however, there was minimal if any gain from using different sorts of features. The features we tried were as follows. (abbreviations used in the tables are listed in italics after the name of the feature). A review of different unigram-like features can be seen in Table 2, whereas a review of higher level combinations of features is presented in Table 3.

1. **Unigrams** (*u*) Standard lower-case unigrams, split on punctuation. These worked fairly

well, and all other features had minimal gain over them.

2. **Bigrams** (*b*) Standard lower-cased bigrams, split on punctuation.

3. **Binarized Unigrams** (*bu*) Binarized lowercase unigrams, where presence of a word in a document earns a feature value of 1, and absence 0. These worked approximately as well as standard unigrams, though they performed better on terrorists.

4. **spellchecking features** (*sp*) Conducts spellcheck on words in a document, using the PyEnchant library. Mildly improved classifiers in conjunction with unigram features.

5. **sentiment features** (*st*) Detected the overall sentiment of a statement, using VaderSentiment (Hutto & Gilbert, 2014). A minor improvement was found with its use.

6. **entity features** (*et*) Many transcripts mentioned named entities. Using the Stanford NER Tagger (Finkel et al, 2005), we count the associated entities. However, this decreased our performance.

7. **hypernym features** (*h*) Using NLTK Wordnet, we generate a list of hypernyms for the first sense of each unigram. These features were developed specifically to help with the Occupation dataset, where many people list oddly specific things. Alas, hypernym features did not seem to help.

8. **synonym features** (*sy*) Synonyms of the first sense of the word, as reported by NLKT WordNet. These tended to boost performance mildly.

9. **bucketed size features** (*sz*) Features indicating how many words were in a particular transcript. These features improved F1 by 0-2% on most datasets.

10. **misspelling features** (*m*) Features indicating the rate at which words were misspelled in a transcript, as well as the total number of misspellings in a transcript. Including these features did not seem to help consistently.

11. **'Ec' feature** (*ec*) A very simple feature, this counts just one thing[1]: how many words in the transcript start in 'ec'. The motivation behind this was because 'the economy' is a very common answer in transcripts, and it is very frequently so misspelled as to preserve only the first two letters in the word.

    The rationale behind this feature was also to give a person an idea of how well custom features worked. Amazingly, even this extremely simplistic feature improved F1 by 1-2%, as seen in table 2.

	u.	**u+ec**	**s**	**bu**
occupation	0.54	0.54	**0.57**	0.54
mip_political	0.83	**0.85**	0.83	0.83
pk_cheney	0.96	0.96	**0.97**	0.96
terrorists	0.66	0.67	0.66	**0.69**

Table 2: Comparing unigram-like features (F1)

5.5 Deep RBM layer

We experimented with using a Bernoulli Restricted Boltzmann Machine (RBM), which is an unsupervised deep learning approach which transforms the feature space into a latent feature space of a different dimension (i.e. a 'hidden layer'). This affected our performance quite negatively; perhaps this is because deep approaches tend to work much better with much more data. We used sklearn's implementation.

6 Results and Discussion

Our results on the model datasets over our different classifiers are presented in figure 6. By and large, over all thirteen datasets and in both dev and test sets, logistic regression performed the best, followed by SVM, and then Random Forests. However, the differences in performance of these models were fairly slight. Our test F1 ranged from .57 on the occupation dataset to .98 on the pk_cheney dataset, with all the pk datasets above .92, and the others below .90. Perhaps because we did not tune our algorithms to any particular dataset, our performance on test and dev sets was equivalent.

Another interesting feature of our process is the high disparity between train and test F1: many classifiers get 100% F1 on the training set. In

[1] pun alert for Hindi speakers

Data set	b+u	b+sy	b+bu	b+sy+h	b+sy+sp	b+sy+st	b+sy+sz	b+sy+m	b+sy+st+sp
occupation	0.53	0.57	0.52	0.57	0.57	0.57	**0.58**	0.57	0.55
mip_political	0.83	0.83	0.83	0.83	**0.85**	0.83	0.83	0.83	**0.85**
pk_cheney	0.96	**0.97**	0.96	**0.97**	**0.97**	**0.97**	**0.97**	0.96	**0.97**
terrorists	0.62	0.63	**0.65**	**0.65**	0.64	0.64	**0.65**	0.64	**0.65**

Table 3: higher level combinations of features (evaluated on F1). The best performance tended to come from bigrams, synonyms, sentiment tags and spelling correction, though the differences were mild.

order to combat this apparent overfitting we experimented with feature selection using metrics based on false positive rate, false discovery rate and Family-wise error rate (via the lovely implementations by the folks at sklearn), all of which decreased our dev error. Going forward, this is something to look into.

Another nice finding has to do with combining datasets. Individually, our model achieved .83 and .84 F1 on each of the mip test datasets, mip_personal and mip_political. However, when we combined them, even though they are asking different questions, our test performance rose to 0.86. And as one recalls mip_personal and mip_political are already datasets combining the respondent's belief as to the first and second greatest personal and political problem. figure 7 illustrates the effect of these four stages of combination on F1 score. The same tends to be true of the occupation dataset, though the highest scores were actually achieved in the current_occupation dataset alone (figure 8).

As a final way of illustrating this, we looked at the past_occupation dataset, which has only 537 training examples. Logistic regression got an F1 score of .31 on the test set when trained on these data. However, when evaluated on the same data but trained on the combined dataset, the F1 soared to .90! These findings highlight the importance of more data.

6.1 Error Analysis

To better understand the sources of error, we did an error analysis of the results that the logistic regression classifier produced on the data set "mip_political". One of the most valuable insights was that for many of the misclassified transcripts, the predicted code is very close to the gold code (Table 4).

For instance, our system classified "military funding" as "national defense (all other)" which is reasonable but not as precise as the gold code

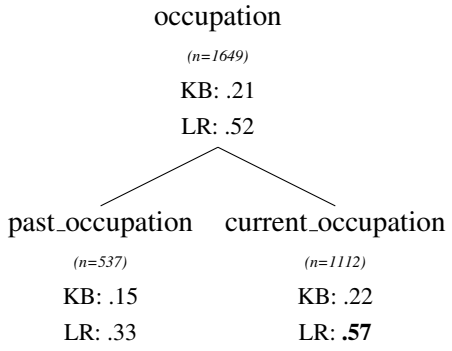

Figure 8: Effect of combining the occupation datasets (F1) is a little ambiguous

Transcript	Predicted	Gold
budget & joblessness	budget	employment
military funding	national defense (all other)	defense spending

Table 4: Misclassifications are often to similar categories

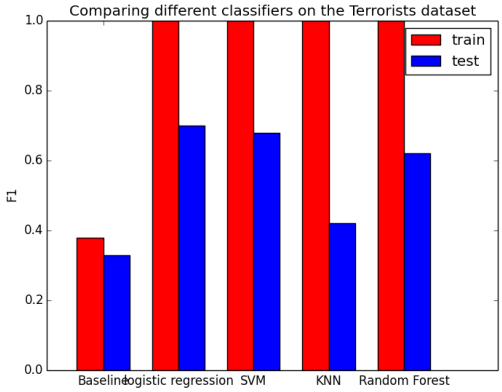

Figure 2: Results on terrorists Dataset

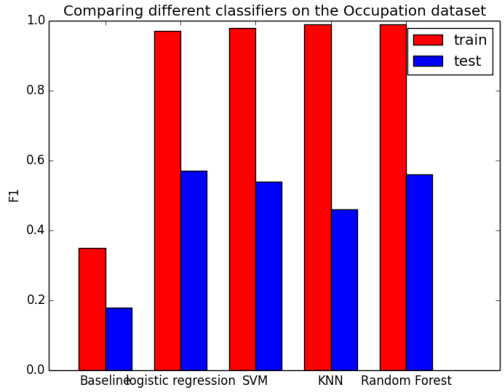

Figure 3: Results on occupation Dataset

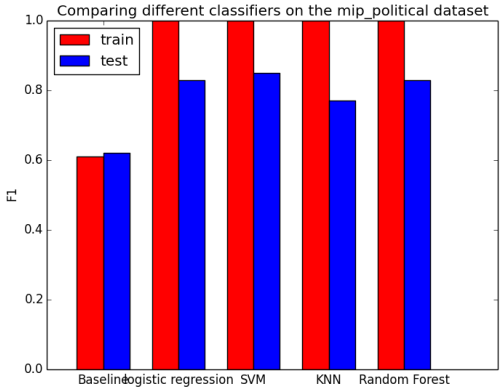

Figure 4: Results on mip_political Dataset

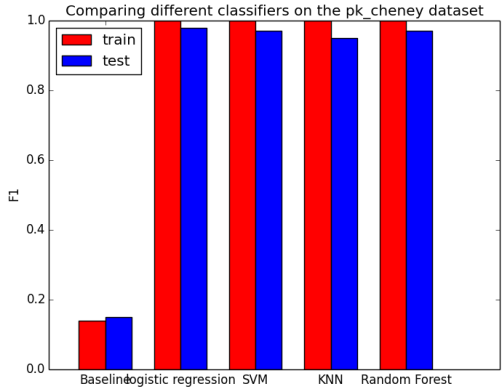

Figure 5: Results on pk_cheney Dataset

Figure 6: Results of different classifiers on our model datasets

"defense spending". We can hypothesize from this analysis that many of our misclassifications are actually not entirely wrong, but failed to predict a more precise code. This analysis also highlights the fact that transcripts fit into multiple codes, which suggests looking into the task of assigning multiple codes to a single transcript, as mentioned in *Future Work*. One way to improve the results would be to mark the codes containing "all other" e.g. "national defense (all other)" as special codes that we only use if no other code within the same super-category reaches a certain confidence. Another way would be to reduce the amount of potential overlap between similar codes.

7 Extension: Self-Weighting Ensemble Models

In addition to the testing above, we experimented with ensemble models, which take the result of several classifiers and combine their results to make a single prediction. Specifically, we developed our own approach, which we call 'self-weighting ensembles', which weights each classifier's prediction for a class with some function of the probability it assigns to that class. However, as these approaches returned negligible if any improvements over straight logistic regression, we detail these approaches and our analysis of their lack of success on these datasets in the appendix.

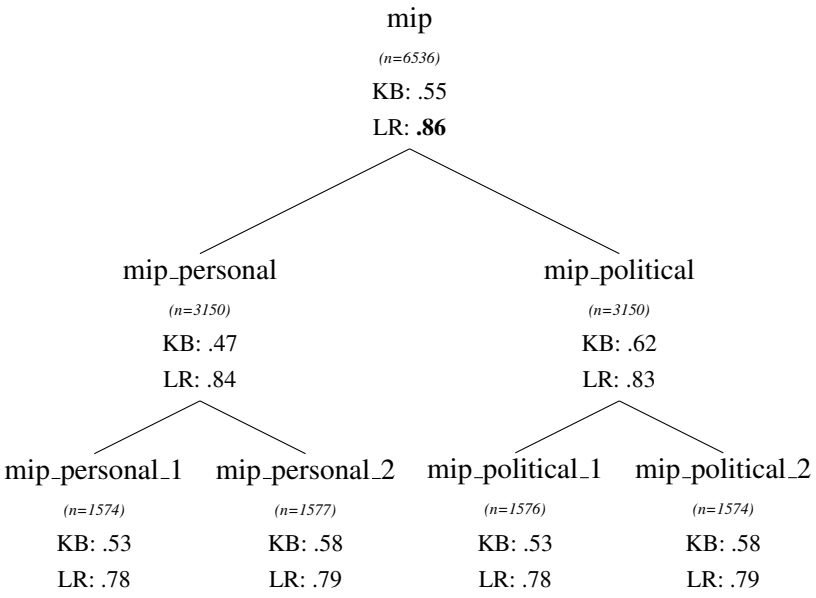

Figure 7: Agglomerating datasets together increases the predictive ability of the models, even if they are responses to different questions. Comparison of F1 scores from Keyphrase Baseline (KB) and Logistic Regression (LR)

8 Future Work

While it is exciting that our programmatic survey of classifiers and features produces such nice results, there are still many things that can be done to improve performance and make a more useful tool for political scientists. Along the lines of improving performance, the most effective next steps are likely to be better preprocessing (for instance, a more nuanced/tuned spellcheck), getting more data, and introducing custom features. The importance of data is highlighted by our better performance on combined datasets, even those asking different questions; the importance of custom features is hinted at by the success of our simplistic 'ec' feature. If we had enough data, it would be lovely to experiment with deep neural networks.

In terms of more nuanced applications, there is work that can be done in the divisions we make when predicting classes. For instance, depending on the particulars of the trend a researcher is studying, she might be interested in only one distinction—perhaps she doesn't care about which of the other 71 classes it is. In this case, training a classifier that specifically discriminates one class from the others might be much more effective. Furthermore, we only predicted one code per transcript; it may be worthwhile to look into prediction of multiple codes. Perhaps most importantly, however, work needs to be done to see how well these techniques perform on supercategories. As noted in the error analysis section, the codes are often very specific, and the mistaken classifications made by our algorithms tend to be very close to the truth. If a political scientist were more interested in general trends, automated techniques could prove even more effective.

Another aproach that we did not experiment with was to look at relations between datasets. For any given transcript from a dataset, we in addition know how that respondent answered for the remaining twelve questions in that survey, information which might be useful for classification. We furthermore have access to demographic information about respondents. We believe that these relations are important to look into, but are skeptical from a philosophical standpoint—if the results of our classifications are used to say something about demographics, and we have used demographics to help make our predictions, any findings of the political inquiry are corrupted. Most likely, any extant trend with any predictive value would be amplified, leading in effect to computational stereotyping.

Even were our system perfect, we have addressed just one tiny part of the intersection of computer and political science. One key element of our work is that it was done on the ANES corpus, a dataset filled with mostly straightforward answers to questions. Even then, our algorithm

had difficulty with nuance in more pointed questions. Much work still is to be done before a system as that brought up by Grimmer (Grimmer & Stewart, 2013) is possible, working on thick political prose with the purpose of gleaning information. The authors, however, are pleased and hopeful, and look forward to whatever advances might come.

9 Recommendations for Survey Data Management

With our high hopes for future work in this interdisciplinary field, we wanted to make sure both groups could together as effectively as possible. In this section, we compile a few recommendations for survey data management from the standpoint of computer scientists which could improve the results of automated machine coding. Most of them center around making things standardized and easily machine readable.

1. **A Standardized Data Format** (including for the mapping of codes to code ids and descriptions), in some raw text format (e.g. .txt, .tsv, .csv, even .xls), rather than Microsoft Word documents. A wonderful format would be .tsv files, with each line containing a transcript followed by the code assigned thereto.

2. **Systems to ensure that transcript data are more clean**, including:

    - If the codes were known in advance of the transcription, it would be wonderful to supply transcribers with a way to provide preliminary labels for a few of the more common classes, such as 'economy' or 'don't know'. This might mean having a convention for abbreviations (e.g. '<dk>' for don't know; currently each transcriber has their own shorthand, inc. "doesn't know", "dk", "<dk>", 'no clue', etc.), or it might mean having check boxes for standard codes, depending on what software transcribers are using.
    - Giving more specific guidelines to interviewers. While most interview transcripts condense what the respondent said, some transcripts include comments on how respondents answered a question. For example, one interviewer wrote "respondent changed her mind and wants to list economic meltdown as most important problem". While a human coder has no problems with extracting "economy" as the most important political problem, our machine classifier has a harder time distinguishing between indirect and direct parts of a transcript. Hence, asking interviewers to take notes in a more standardized way without loss of information could improve automated coding results.

References

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

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling.* Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370.

Bird, Steven, Edward Loper and Ewan Klein. 2009. *Natural Language Processing with Python.* OReilly Media Inc.

Tsytsarau, Mikalai, Themis Palpanas 2012. *Survey on Mining subjective data on the web* Data Mining and Knowledge Discovery Volume 24 Issue 3, pp. 478-514

Grimmer, J., and B. M. Stewart. 2013. *Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.* Political Analysis 21.3 pp.267-97.

King, Gary, and Will Lowe. 2003. *An Automated Information Extraction Tool for International Conflict Data with Performance as Good as Human Coders: A Rare Events Evaluation Design.* International Organization 57.03

DeBell, Matthew. 2003. *Harder Than It Looks: Coding Political Knowledge on the ANES.* Political Analysis 21.4, pp.393-406.

Stanford University and the University of Michigan. 2008. The American National Election Studies (ANES; www.electionstudies.org).

Bowman, Samuel R.; Christopher Potts; and Christopher D. Manning. 2015. *Learning distributed word representations for natural logic reasoning.* Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches: Papers from the 2015 AAAI Spring Symposium, 10-13. AAAI Publications.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. 2011. *Domain adaptation for large-scale sentiment classification: A deep learning approach.* Proceedings of the 28th International Conference on Machine Learning (ICML-11).

Hutto, C.J. & Gilbert, E.E. 2014. *A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text.* Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI

George A. Miller 1995 *WordNet: A Lexical Database for English* Communications of the ACM Vol. 38, No. 11: 39-41.

Roberts, Margaret E., et al. 2014 *Structural Topic Models for OpenEnded Survey Responses.* American Journal of Political Science 58.4 pp. 1064-1082.

APPENDIX

A Self-Weighting Ensembles

The idea of the self-weighting ensemble is to weight each prediction by the confidence of the prediction. This way, if there are certain instances in which a certain classifier might do much better/predict with much more certainty, its vote will be given more credence. The self-weighting ensemble weights each classifier's prediction by a function of its own predicted probabilities (hence 'self-'), and then sums the result. Specifically, we pass the predictions through a nonlinear transformation which favors predictions in certain regions of the probability space. Of the nonlinearities we experimented with, the following are of note:

1. **Sigmoid Nonlinearity**: Weight each prediction by the sigmoid function, e.g. $p' = \frac{1}{1+e^{-p}}$. This transformation favors predictions that are closer to 0 or 1, and gives less weight to predictions closer to 0.5. (Note that we first transform the probability into the interval [0,1])

2. **Rectified Linear Unit (ReLU) Nonlinearity** Weight predictions by $p' = max(.5, p)$. This favors positive predictions.

3. **Inverse Sigmoid** Weight predictions by the inverse sigmoid (with concavity reversed from that of the normal sigmoid). This favors classifiers which predict closer to 0.5, rather than the extremes.

4. **constant** Simply sum probabilities together

5. **binarized** One classifier, one vote: whatever class was predicted by a majority wins.

From our preliminary analysis, the ensemble of Logistic Regression with Random Forests performed well with most nonlinearities (with constant and sigmoid slightly in the lead), outperforming logistic regression by .2 % F1 to 3% F1 on certain datasets. The latter case seemed anomalous, however, and it seemed that the ensembles were not performing consistently better than Logistic Regression (in some cases, they were much worse). We hypothesize that they did not perform well because many different classifiers were getting the same things wrong. This could be a symptom of the low dimensionality of our data, in that there are not too many interesting ways to separate them.