Speeding up the manifesto project: Active learning strategies for efficient automated political annotations

Felix Biessmann[*], Philipp Schmidt[†]

January 1, 2018

Abstract

The Manifestoproject Corpus is an exceptional data source for political education as it combines political texts with valuable annotations by human experts. However the amount of data human annotators can label is very limited compared to the ever increasing amount of political texts published in manifestos, news media and social networks. The discrepancy between labeling budget and data that needs to be labelled highlights the necessity of automated annotations by means of machine learning (ML).

When only a small fraction of the available data can be labelled in order to train an ML model, the priority of which samples should be labelled first is important: Some samples are easy to classify - those should not be labelled with high priority, as the classifier will not learn much from them. For these samples it is relatively safe to have them labelled automatically by an ML model without wasting the time of human annotators. Other data points are difficult for the ML model; when labels for those are obtained first, the model will reach its optimal classification performance faster.

In this study we leverage this insight, known in the ML community as active learning. We present offline experiments on the manifesto corpus showing that active learning strategies significantly speed up training of ML models for manifesto code annotations. This shows the potential of ML methods as assistive technology for political science. To demonstrate the benefits of this approach we present an implementation of a web based active learning annotation system that can be readily used for speeding up the manifesto annotations as well as annotations of other political texts.

1 Introduction

Automated political text analysis is an increasingly popular field of research. The reasons for this include research in political sciences, political education and also the need for unbiased media consumption. All of these applications require assistive technology for automated political analysis if one wants to keep up with the ever growing amount of media content published. For training assistive machine learning technology large annotated data sets are needed.

[*]felix.biessmann@gmail.com
[†]schmidtiphil@gmail.com

In this context often labelling resources are limited and cannot be scaled easily to increasing annotation demands. One example is a time limit on the annotations, when parties release their policies only shortly before the elections. Often these texts are too long to be read by one single person before the elections [11]. In addition to relying on experts to interpret these programs, it can be helpful to use models trained on previously annotated data to assess the policies proposed in the party programs. This can help to get a better overview of the policies especially when new parties enter the political stage, see e.g. [7]. Another scenario in which the labelling budget is limited is annotation of texts from online media. The ever increasing volume of content published online does not allow for an analysis of all texts by human experts. Thus automated text analysis with models trained on already annotated data can be helpful. Yet another example in the context of political sciences is that often in political text annotations there is an annotator bias in the annotations which leads to inconsistent or noisy annotations. Measuring and fixing those biases requires at least three labels per data point. Multiplying the required annotations by a factor of three is however costly. If labelling resources are limited, it can make sense to not spent all annotations to get as many samples as possible annotated but rather to try and also use some annotations to increase the number of data points for which more than one annotation is available.

The field of machine learning has focused on this case of limited labelling budget in a branch called *active learning*, for an overview see e.g. [14]. One example of an active learning problem is similar to the above sketched scenario: A large amount of texts needs to be labeled with a limited labelling budget such that a classifier trained with the limited labeled examples is as good a possible in terms of classification performance on a test set. In offline experiments the quality of the model is evaluated against a classifier trained on a data set in which all data points are labelled.

Active learning tries to give high priority in the annotation queue for those samples for which the classifier is most uncertain and to give a lower priority to samples which are easy to classify correctly. Intuitively speaking the reason for this is that classifiers learn more from hard examples than they learn from easy ones. Mathematically speaking the reason is that the gradient of the error function that is minimized during training is small for correctly classified examples and larger for incorrectly classified samples or samples where the model is uncertain.

In this study we employ several active learning strategies in order to assess the potential of active learning for automating political annotations. We use the latest version of the Manifesto Project Corpus as data set and run offline experiments in which we compare a random sampling strategy for annotation of increasing fractions of the total data set with active learning sampling strategies.

This manuscript is structured as follows: In the next section a brief summary of some related work on automated political text analysis is provided, then section 3 describes the data set and the methods for preprocessing the text data; in section 4 the machine learning model is discussed and in subsection 4.2 various active learning strategies are presented. Finally section 5 summarizes the results of the experiments.

2 Related Work

Throughout the last years automated content analyses for political texts have been conducted on a variety of text data sources (parliament data blogs, tweets, news articles, party manifestos) with a variety of methods, including sentiment analysis, stylistic analyses, standard bag-of-word (BOW) text feature classifiers and more advanced natural language processing tools. For an overview we refer the reader to [5, 2].

The work closest related to this study is using machine learning classification models to identify political affiliation or bias on political speeches or manifesto data. In [16] the authors extracted BOW feature vectors and applied linear classifiers to predict political party affiliation of US congress speeches. Similar work in [6] uses data from the Canadian parliament and the European parliament. Other work has focused on developing largely unsupervised methods for predicting political bias. Two popular methods are WordFish [15] and WordScores [8], or improved versions thereof, see e.g. [10]. These approaches have been very valuable for *a posteriori* analysis of historical data but they do not seem to be used as much for analyses of new data in a predictive analytics setting.

Taken together, there is a large body of literature on automated analysis of political texts. Except for few exceptions most previous work has focused on binary classification or on assignment of a one dimensional policy position (often corresponding to left vs right). Data sets such as the manifesto project allow for more fine grained analyses of political bias. Models trained on such data can give valuable insights in political texts in a predictive setting see e.g. [12, 3]. While data set as the manifesto project allow to train classifiers that can give more detailed insight into political bias, they also impose a significant effort on the annotators.

Despite the challenges associated with acquiring annotated data for training automated political analysis models, to the best of our knowledge there is no work in the political sciences that tries to leverage active learning strategies for speeding up the label acquisition process. While this could be due to the fact that many annotation efforts aim at an exhaustive annotation of the entire data set and thus would not profit from an active learning approach, we argue that there are many scenarios, as outlined in section 1, that suffer from the problem of a limited annotation budget and thus can profit from active learning.

3 Data Sets and Feature Extraction

This section describes the data set and preprocessing applied to the Manifesto Project Corpus data, we used the latest version available when running experiments, version 2017b.

3.1 Data

Annotated political text data was obtained from all manifesto texts of parties in Germany made available through the Manifesto Project [9]. Each sentence or in some cases also parts of a sentences is annotated with one of 56 political labels. Examples of these labels are *pro/contra protectionism, decentralism, centralism, pro/contra welfare*; for a complete list and detailed explanations

cmp_code	label	counts
503	social justice +	4086
411	infrastructure +	3085
501	environmentalism +	2775
303	gov-admin efficiency +	2580
403	market regulation +	2513
504	welfare +	2418
201	freedom/human rights +	2410
701	labour +	2199
506	education +	2064
202	democracy +	1918
305	political authority +	1590
107	internationalism +	1574
408	economic goals	1298
706	non economic groups +	1224
606	social harmony +	1137
502	culture +	1130
605	law and order +	1117

Table 1: List of valid labels and Manifesto Project Codes above 1000 counts.

on how the annotators were instructed see [1]. In total this resulted in 48148 political statements. In order to obtain training data that would lead to reliable predictions in the experiments we discarded categories that had less than 1000 observed labels, which resulted in the 17 labels listed in subsection 3.1 and reduced the data set to 35,118 samples, or 73% of the original data set.

The reason we needed to discard the less frequent labels was merely because in the experiments we subsampled the data heavily in order to simulate cases in which only very few sentences could be labeled. This required us to discard rare labels as those are very likely to not be observed at all when subsampling only around 10% of the data.

3.2 Bag-of-Words Vectorization

Strings of each semantic unit (quasi sentence) were tokenised and transformed into bag-of-word vectors as implemented in scikit-learn [13]. We used the standard HashingVectorizer. The text of each semantic unit is transformed into a vector $\mathbf{x} \in \mathbb{R}^d$ where $d = 10^{18}$ is the size of the dictionary (here: number of hash buckets); the wth entry of $\mathbf{x}$ contains the count of the wth feature. In previous experiments we have tried several options for vectorizing the texts, including term-frequency-inverse-document-frequency normalisation, n-gram patterns up to size $n = 3$, several cutoffs for discarding too frequent and too infrequent words and also most standard natural language processing procedures such as stemming or lemmatization, for some of these experiments we refer to [3]. These experiments showed that rather modest preprocessing, lowercasing and bigrams, resulted in best generalization performance. Hence in the current experiments we only used lower casing and unigram features, which were only slightly worse than bigrams.

4 Classification Model and Training Procedure

Bag-of-words feature vectors were used to train a multinomial logistic regression model. Let $y \in \{1, 2, \ldots, K\}$ be the true label, where K is the total number of labels and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K] \in \mathbb{R}^{d \times K}$ is the concatenation of the weight vectors $\mathbf{w}_k$ associated with the kth party then

$$p(y = k|\mathbf{x}, \mathbf{W}) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \qquad \text{with } z_k = \mathbf{w}_k^\top \mathbf{x} \tag{1}$$

We estimated $\mathbf{W}$ using stochastic gradient descent with 5 epochs through the training data set. The optimization function was obtained by adding a penalization term to the negative log-likelihood of the multinomial logistic regression objective and the optimization hence found the $\mathbf{W}$ that minimized

$$L(\mathbf{W}, \mathbf{x}, \gamma) = -\log \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} + \gamma \|\mathbf{W}\|_F \tag{2}$$

Where $\| \ \|_F$ denotes the Frobenius Norm and γ is a regularization parameter controlling the complexity of the model. The regularization parameter was optimized on a log-scaled grid from $10^{-4,\ldots,4}$. The performance of the model was optimized using the negative log likelihood, but we also report all other standard measures for each class, in particular

- Accuracy $\frac{\text{correct classifications}}{\#samples}$
- Precision $\frac{TP}{FP+TP}$
- Recall $\frac{TP}{TP+FN}$
- F1 score $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

where TP and FP stands for true positives and false positives, respectively.

4.1 Model Selection and Evaluation

For model evaluation we kept 10% of the entire training data separate in all experiments. For model selection, or hyper parameter optimization we only considered the regularization parameter as described in section 4. All other hyper parameters related to preprocessing the data were kept fixed based on previous experiments reported in [3]. This restriction was made to speed up the experiments, as each training run involved hyper parameter optimization, which comes at a significant computational cost. All hyperparameters were optimized using 2-fold cross validation on the training set.

4.2 Active Learning Strategies

We employed three different active learning strategies which we will briefly outline below, for a detailed review on the topic and various strategies we refer to [14].

Random Baseline In order to simulate the standard procedure for sampling that does not take into account the uncertainty of the classifier we used random sampling from the pool of to-be-labeled examples.

Uncertainty Sampling Uncertainty sampling queries samples $\mathbf{x}_i$ for which a classifier's top prediction is least confident

$$\mathbf{x}_i = \underset{i,k}{\operatorname{argmax}} \, (1 - p(y = k|\mathbf{x}_i, \mathbf{W})) \tag{3}$$

This strategy only takes into account the top prediction and hence could loose information from less likely classes, according to the model.

Entropy Sampling Entropy sampling queries samples $\mathbf{x}_i$ for which a classifier's prediction for all classes is least confident

$$\mathbf{x}_i = \underset{i}{\operatorname{argmax}} \sum_k p(y = k|\mathbf{x}_i, \mathbf{W}) \log(p(y = k|\mathbf{x}_i, \mathbf{W})) \tag{4}$$

This variant takes into account predictions for all classes and intuitively speaking might pick up too much information from noisy predictions for unlikely classes.

Margin Sampling Margin sampling queries samples $\mathbf{x}_i$ for which a classifier's prediction for the top two predicted classes is minimal

$$\mathbf{x}_i = \underset{i}{\operatorname{argmin}} \, (p(y = k_1|\mathbf{x}_i, \mathbf{W}) - p(y = k_2|\mathbf{x}_i, \mathbf{W})) \tag{5}$$

where k_1, k_2 are the classes that are most likely and second most likely, respectively, under the current model. This variant is a compromise between the uncertainty sampling and the entropy sampling strategy.

4.3 Offline Experiment Setting

We used the above active learning strategies to simulate a scenario in which a limited budget of labelling resources is available and the task is to train a model as best as possible on this labelling budget. The model performance was measured on a separate held out test data set. The best performance of a given model class was measured by performing model selection and training on all of the training data and testing on the test set. The active learning strategies were then compared by performing model selection and training on only a fraction of the training data, in particular 1%, 10%, 20%, . . . , 100%. For each subsampling of the data we performed all four sampling strategies. Sampling was performed using the last model on all samples that were not yet labeled and the newly sampled data points were joined with the already sampled data points. This setting allowed to estimate with which fraction of the entire training data each respective strategy yielded the best model, relative to the model trained on all training data. In order to obtain robust results we ran each experiment 100 times, meaning 100 times training and testing using each sampling strategy for each of the considered fractions of the training data.

manifesto code	precision	recall	f1-score	support
107	0.60	0.48	0.53	774
201	0.51	0.55	0.53	1194
202	0.63	0.57	0.60	983
303	0.35	0.41	0.38	1251
305	0.46	0.59	0.52	783
403	0.52	0.48	0.50	1281
408	0.34	0.17	0.23	628
411	0.39	0.60	0.47	1535
501	0.61	0.55	0.58	1380
502	0.65	0.41	0.50	587
503	0.46	0.52	0.49	2083
504	0.34	0.56	0.43	1179
506	0.63	0.48	0.54	1026
605	0.56	0.44	0.49	576
606	0.63	0.27	0.38	561
701	0.59	0.39	0.47	1123
706	0.45	0.26	0.33	615
avg / total	0.50	0.48	0.48	17559

Table 2: Precision, recall, F1 score and number of instances per class.

5 Results

We first report some results of a model that was trained on the entire training data, in order to illustrate the performance of the best performing model under the given model class. This will be the bar against which the active learning experiments will be compared. Afterwards we present the results of the active learning experiments in which we compare the strategies listed in subsection 4.2 with a random baseline.

5.1 Baseline Model

Example results for a baseline model trained on half the available training data is shown in Table 2. On average an F1 score of 0.48 was obtained with a simple linear logistic regression model and unigram features. This is comparable to results previously obtained on the complete set of manifesto codes and a model trained with exhaustive model selection on all relevant model and preprocessing hyperparameters. Also the result is comparable with the accuracy obtained in previous similar experiments performed on the same data [12].

5.2 Active Learning Results

We ran 100 offline experiments on all data for all active learning strategies and plot the median and 5th/95th percentile of the accuracies obtained with the respective model trained on a given fraction of the training data in Figure 1. The results demonstrate that active learning clearly achieves superior performance compared to random sampling. In the large majority of cases, active learning resulted in a model that was indistinguishable from the best model that could

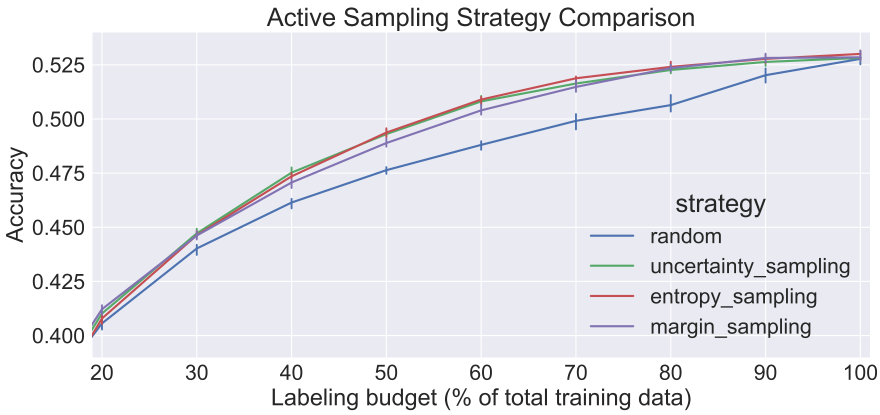

Figure 1: Comparison of active learning strategies, see subsection 4.2, with random baseline (blue), shown are the median and the 5th/95th percentile across 100 repetitions. While there are no distinct differences between the active learning strategies, all active learning strategies are significantly superior to randomly sampling from the pool of unlabelled data points. With less than 80% of the total training data active learning is able to learn models that have the same performance as the best possible model, which was trained on all data.

have been obtained when training on all available training data with less than 80% of the total training data. Random sampling, the baseline strategy, required 100% of the training data in order to achieve the same performance. We found active learning in our experiments to be able to reach over 90% of the best model's performance with less than 40% of the available training data. Active learning based sampling often required less than 20% fewer data points than random sampling to achieve the same performance.

We did not observe in our experiments a clear distinction between the different active learning strategies. This could be related to the fact that we discarded rare labels, which led to a rather small amount of classes. Hence the differences between the active learning strategies and their respective advantages for a large number of labels did not help as much as they would have with more labels and an even more skewed label distribution.

6 Conclusion

Our offline experiments show that active learning can be useful for speeding up annotation of political texts. The results of this study demonstrate that active learning strategies can yield models for automated political analysis that are as accurate as the best model, meaning a model trained on all available training data, with less than 80% of the entire training data in 95% of the experiments. Random sampling in contrast achieves only the best performance when trained on 100% of the data. This highlights the potential of using information of the model uncertainty when deciding which samples to annotate next when the annotation budget is limited.

As we deliberately restricted the complexity of the model class to a linear classifier and simple unigram bag of words features, it is very likely that the best performing model can be improved. This does not affect the conclusion that active learning can speed up the annotation process. Our restrictions were made deliberately and merely to speed up experiments.

All code for this study is publicly available through github [4].

Acknowledgements

We thank Giovanni Zappella for helpful feedback on active learning matters. We thank Jirka Lewandowski and Pola Lehmann for helpful discussions on data, modelling and support when using the public Manifesto API.

References

[1] Manifesto codebook `https://manifesto-project.wzb.eu/information/documents?name=handbook_v4`.

[2] Isa Maks Bertie Kaal and Annemarie van Elfrinkhof, editors. *From Text to Political Positions: Text analysis across disciplines*. John Benjamins, 2014.

[3] Felix Bießmann. Automating political bias prediction. *CoRR*, abs/1608.02195, 2016.

[4] Felix Biessmann and Philipp Schmidt. Active learning for the manifesto project. *https://github.com/felixbiessmann/active-manifesto*, 2017.

[5] Justin Grimmer and Brandon M. Stewart. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 2013.

[6] Graeme Hirst, Yaroslav Riabinin, Jory Graham, and Magali Boizot-Roche. Text to ideology or text to party status? In Isa Maks Bertie Kaal and Annemarie van Elfrinkhof, editors, *From Text to Political Positions: Text analysis across disciplines*, pages 47–70, 2014.

[7] Steffen Kühne, Oliver Schnuck, and Robert Schöffel. Der computer sagt: Jamaika. *https://web.br.de/interaktiv/wahlprogramm-analyse-bundestagswahl/*, 2017.

[8] Michael Laver, Kenneth Benoit, and John Garry. Extracting policy positions from political texts using words as data. *American Political Science Review*, 2003.

[9] Pola Lehmann, Theres Matthieß, Nicolas Merz, Sven Regel, and Annika Werner. *Manifesto Corpus*. WZB Berlin Social Science Center., `https://manifestoproject.wzb.eu/information/documents/corpus`, 2015.

[10] Will Lowe, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. Scaling policy preferences from coded political texts, 2009.

[11] Nicolas Merz. Alle wahlprogramme lesen? dauert nur 17 stunden. *http://www.zeit.de/politik/deutschland/2017-08/bundestagswahl-wahlprogramme-parteien-computeranalyse*, 2017.

[12] Nicolas Merz, Sven Regel, and Jirka Lewandowski. The manifesto corpus: A new resource for research on political parties and quantitative text analysis. *Research & Politics*, 3(2):2053168016643346, 2016.

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

[14] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[15] Jonathan B. Slapin and Sven-oliver Proksch. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 2008.

[16] Bei Yu, Stefan Kaufmann, and Daniel Diermeier. Classifying party affiliation from political speech. *Journal of Information Technology & Politics*, 5(1):33–48, 2008.