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

December 19, 2017

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). Taking into account label consistency across human experts, which is often not perfect and thus requires to collect at least three labels per data point, this discrepancy is even worse.

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

2 Data Sets and Feature Extraction

2.1 Data

Annotated political text data was obtained from all manifesto texts of parties in Germany made available through the Manifesto Project [2]. Each sentence

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.

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 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 2.1.

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.

2.2 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 [3]. 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 [**?**]. These experiments showed that rather modest preprocessing, lowercasing and bigrams, resulted in best generalization performance. Hence in the current ex-

periments we only used lower casing and unigram features, which were only slightly worse than bigrams.

3 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 quasi-newton gradient descent. 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.

3.1 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 3. All other hyper parameters related to preprocessing the data were kept fixed based on previous experiments reported in [**?**]. 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.

3.2 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 [4].

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

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

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

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

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.

3.3 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.

4 Results

4.1 Baseline Model

In order to Example results for a baseline model trained on half the available training data is shown in Table 2.

4.2 Active Learning Results

Basically: it works, see also Figure 1.

5 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 when using active learning can yield models that are as accurate as the best model, meaning a model trained on all available training data, with less than XX% of the entire training data in 95% of the experiments. Random sampling in contrast achieves only the best performance when trained on XX% 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.

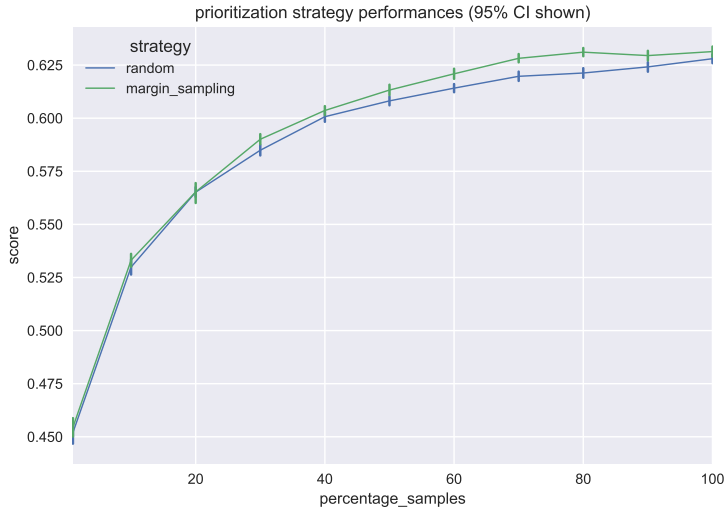

Figure 1:

Acknowledgements

References

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

[2] 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.

[3] 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.

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