A comparison of methods in political science text classification: Transfer learning language models for politics[*]

Zhanna Terechshenko[†], Fridolin Linder[‡], Vishakh Padmakumar[§], Fengyuan Liu[¶], Jonathan Nagler[‖], Joshua A. Tucker[**], Richard Bonneau[††]

November 5, 2020

Abstract

Automated text classification has rapidly become an important tool for political analysis. Recent advancements in NLP enabled by advances in deep learning now achieve state of the art results in many standard tasks for the field. However, these methods require large amounts of both computing power and text data to learn the characteristics of the language, resources which are not always accessible to political scientists. One solution is a transfer learning approach, where knowledge learned in one area or source task is transferred to another area or a target task. A class of models that embody this approach are language models, which demonstrate extremely high levels of performance. We investigate the performance of these models in the political science by comparing multiple text classification methods. We find RoBERTa and XLNet, language models that rely on the Transformer, require fewer computing resources and less training data to perform on par with – or outperform – several political science text classification methods. Moreover, we find that the increase in accuracy is especially significant in the case of small labeled data, highlighting the potential for reducing the data-labeling cost of supervised methods for political scientists via the use of pretrained language models.

[*]Z.T. and F.L. are co-first authors, and contributed equally to this work by running the analyses and writing the first drafts of the paper. V.P. and F.L. assisted with the empirical analyses. F.L., Z.T., and R.B. produced the original research design, and J.N. and J.T. contributed to revisions of the research design. F.L., Z.T., J.N., J.T. and R.B. all contributed to the editing of mansucript.

[†]Center for Social Media and Politics, New York University, New York, NY 10012.

[‡]Siemens Mobility, Munich, Germany.

[§]NYU Center for Data Science, New York, NY 10012.

[¶]Department of Computer Science, New York University, Abu Dhabi, UAE.

[‖]NYU Department of Politics and Center for Social Media and Politics, New York, NY 10012.

[**]NYU Department of Politics and Center for Social Media and Politics, New York, NY 10012.

[††]NYU Department of Biology and CS and Center for Social Media and Politics, New York, NY 10012.

1 Introduction

Supervised text classification has become an integral part of many areas of research in political science, and the social sciences more broadly. The categorization of unstructured text into topical, relevance, or stance categories has become an increasingly important step in the annotation of large amounts of unstructured data for more detailed analysis (Grimmer and Stewart, 2013). The ability to classify unstructured text data with high levels of accuracy, however, is often expensive as it requires relatively large amounts of human-labeled data and significant computing power. However, in many cases, labeled data from a related task is already available and costly data collection can be avoided (or the amount of additional labeled data required can be reduced). For instance, categorization of political text into issue/topic categories is a problem that occurs in many research applications (Lowe and Benoit, 2013). Large data collection efforts such as the Comparative Agendas Project (CAP) (Baumgartner et al., 2006) or the Comparative Manifestos Project (Volkens et al., 2018) could in principle provide large amounts of data for training general classifiers that could then by applied to related smaller datasets, provided the initial coding scheme is sufficiently general or particularly applicable to the task at hand.[1]

One potential solution to this problem is to identify methods that exhibit consistently high performance across tasks, as well as methods that can provide reliable feature extraction across related but non-overlapping task-dataset pairs. This type of learning is referred to in the computer science literature as *transfer learning*. While transfer learning as a category of machine learning has been studied for decades, recent innovations in the architecture of deep learning models – as well as the availability of the computational resources to fit models with multiple hundreds of millions of parameters – have led to increased focus on transfer learning in conjunction with natural language processing tasks. Neural transfer learning models, such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) and

[1]An alternative approach that is already widely used in political science are topic models. Even though they have the advantage of not requiring labeled data, such unsupervised models are not guaranteed to produce categories that map onto a schema that is useful to the researcher. Additionally, topic models have been found to be relatively unstable and unreliable in some more challenging transfer learning settings (Chuang et al., 2013; Denny and Spirling, 2018).

Generalized Autoregressive Pretraining for Language Understanding (XLNet) (Yang et al., 2019), have shown high levels of accuracy across different natural language processing applications, but relatively little is known about the applicability of these models across different subject domains generally, and political science in particular. Here we aim to provide political scientists with the guidance for text classification across diverse tasks.

In this manuscript, we provide a brief overview of common methods for supervised text classification and discuss transfer learning via pretrained language models as a potential solution for text classification problems in political science. We put this claim to test by comparing the performance of traditional machine learning and several transfer learning models across four data sets that represent typical political text classification tasks. We also conduct an analysis on a more difficult classification task that involves training the classifier on labeled data from one data set (here congressional bills from the *Comparative Agendas Project*) and testing it on another (here, *New York Times* headlines). In this setting, we simulate the situation described above, where there is a constraint on data availability for a task in question but where labeled data from a related task is available. Finally, we provide software for both cross-validation and for implementation of the model that achieves the highest accuracy in our analyses.

We find that transfer learning models that use the Transformer architecture perform consistently well (on par or outperform other models). We also find that the increase in accuracy is especially significant in data sets with the smallest amount of labeled data. In the harder classification task described above, these models again demonstrate much higher accuracy than traditional machine learning models. However, this accuracy is much lower than in cases where the models were trained and tested on the same data set. Taken together, our results suggest that transfer learning models using the Transformer architecture perform consistently well, and while they cannot remove the need for labeled data altogether, they can reduced the amount of labeled data required for a given task. Furthermore, these models are readily available for several languages, relatively easy to use and very fast, which in combination with high levels of accuracy suggests that these models represent a powerful tool for political science research moving forward.

2 Task Specific vs Transfer Learning

Before comparing transfer learning methods to traditional machine learning models, it is important to understand the difference between these models. The majority of common machine learning methods are applied in a task-specific manner where models are trained to address isolated tasks, which they learn from scratch using the training data fed into these models. Transfer learning describes a class of tasks in machine learning where knowledge or data in one area is transferred, or re-used, in another area. Most of these tasks fall under one of four classes, based on what is transferred from the source domain/task to the target: instance transfer, feature representation transfer, parameter transfer, and relational knowledge transfer (Pan and Yang, 2010). The main approach that we are interested in here is that of *feature representation transfer*, where the goal is to find a "good" feature representation that reduces the difference between the source and the target domains and therefore the error of classification models. Thus, the basic goal is to develop a text representation from the source domain that can be reused for the target task.[2]

The most widely used methods of feature representation transfer are word embeddings (Mikolov et al., 2013; Pennington et al., 2014), which aim to learn a vector representation for each word in a vocabulary based on the (unsupervised) context in which each word appears. Word embeddings have already been adopted in political science research (Spirling and Rodriguez, 2019; Rudkowsky et al., 2018), and are meant to capture patterns in language and word use common to most/all texts. Usually, word embeddings are trained using very large corpora of language in an unsupervised manner, in order to produce meaningful representations (e.g. Spirling and Rodriguez (2019) find more consistently meaningful representations from pretrained vs. local embeddings, most likely due to the much larger size of the corpora that are used to produce these pretrained embeddings). That is, information about the mean-

[2]See Pan and Yang (2010) for a more extensive overview and categorization of different transfer learning problems and proposed solutions.

ing of words is learned from text that is unrelated to the supervised text classification problem at hand.

So called neural language models (Devlin et al., 2018; Howard and Ruder, 2018; Radford et al., 2018, e.g.) are motivated in a similar fashion. Instead of simply learning vector representations of words or phrases that will remain constant over each occurrence of the word in the text corpus, as embeddings do, these models learn to produce meaningful vector embeddings of sequences of text. That is, these more advanced neural models do not just embed words in isolation, but additionally encode the order and relationships of words in sequence instances, making them substantially more powerful at representing text for supervised classification (Peters et al., 2018). Having been pretrained on very large-scale corpora these models are able to transfer this knowledge to initialize and then train another model to perform extremely well on a specific NLP task. In the next few sections we provide a brief review of text representations for supervised learning, including word embeddings, as well as language models with the focus on the models we test in this paper, namely ULMFIT (Howard and Ruder, 2018), RoBERTa, and XLNet (Yang et al., 2019).

3 Methods for Supervised Learning from Text

Here we briefly review a subset of the large variety of methods to represent text as inputs to a learning method, with a focus on methods appropriate for more common politics text classification tasks.

3.1 Word n-grams

Word n-grams are a commonly used way of representing text in a supervised learning task. Each document is split into tokens (unigrams, or single words that can be normalized by tokenization or lemmatization) or short sequences of tokens (e.g. bi- or tri-grams). The document is then represented by a fix length vector of counts or other measures of how often each term of a global vocabulary (over all documents) occurs in the given document. Word n-grams have several advantages: They are easy to calculate and don't require additional resources but the data at hand. Depending on the supervised algorithm used, word n-grams provide interpretable features, i.e. terms that are predictive of the outcome can be understood and models can be diagnosed. The main disadvantage of word n-grams is sparsity, the phenomenon of not observing enough data (for a subset of words) in a corpus to model the true distribution of language. This sparsity can lead to a problematic computational burden in some situations: models have to be fit to relatively large matrices (especially in comparison to word and document embeddings).

3.2 Character n-grams

Character n-grams differ from the word n-grams in the way the document is split into tokens. Instead of breaking up documents at the word boundaries, each word is split into character groups of size n. The document is then represented by a frequency vector (or a transformation) over a global character n-gram vocabulary. Character n-grams can be advantageous in situations where words are spelled in unusual ways, for example with text from social media platforms. In a word n-gram representation the terms '#HillaryClinton' and 'Hillary Clinton' would be a completely separate dimension. In a character n-gram representation, however, many of the sub-word character grams would overlap between the two terms. Character n-grams lead to large feature spaces as well and can therefore cause computational issues.

3.3 Embeddings

An embedding is simply a representation of a document (or portion of a document) as a fixed length vector. The most widely known form of embeddings are word embeddings that are combined into a document embedding. A k-dimensional space is defined a-priori and words are represented in this space by their co-occurrence with other words. An intuitive example for a word embedding is a lower dimensional representation of a word co-occurence matrix. Say the meaning of a word is represented by the number of times it co-occurs with a set of words in a fixed vocabulary within a specified corpus of documents (or sentences or other small 'contexts'). A lower-dimensional representation of each word can then be obtained by factorization of this co-occurence matrix. A method to do so that is common in the social sciences would be principal component analysis or factor analysis. In practice, however, word embeddings are 'trained' using shallow neural networks such as in `Word2Vec` (Mikolov et al., 2013) or `GLOVE` (Pennington et al., 2014). In order to obtain a document embedding, the document can be split up into words and the vectors for each word can be merged or pooled by, for example, taking the average (mean pooling) or the maximum (max pooling) of each dimension of the embedding.

Word embeddings offer many theoretical advantages over normal bag-of-words or bag-of-n-gram representations. If they are trained on a sufficiently large corpus, words with similar meanings should have representations that are close to each other (Firth, 1962). This leads to reducing sparsity of the representation of the document while still producing similar representations for similar documents. One limitation of standard word embedding techniques such as `Word2Vec` or `fastText` is that these methods, along with all bag-of-words methods, do not take the word context into account and thus fail to distinguish between the words that have different meanings depending on the context in which these words appear.

3.4 Language Models

A language model is a probability distribution over a sequence of words in the document and it can be used to produce a vector representation that considers all features of said document. That is, the models take account of not only the 'bag of words' of the document, as in n-gram and word embedding based representations, but also the order of words, punctuation and grammar. By considering word order, the model is capable, at least in principle, of encoding not only the semantic meaning derived from word frequencies, but also the syntactic structure and therefore meaning that is a consequence of grammar (e.g. negation, questions, subject-verb-object relations). There are multiple ways of training such language models, but neural network based language models have gained increased attention across multiple disciplines recently for their impressive performance on a variety of tasks involving natural language (Devlin et al., 2018; Howard and Ruder, 2018; Radford et al., 2018; Yang et al., 2019, e.g.). We focus on three of these models: Universal Language Model Fine-Tuning (ULMFIT) (Howard and Ruder, 2018), A Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019), and Generalized Autoregressive Pretraining for Language Understanding (XLNet) (Yang et al., 2019). Both RoBERTa and XLNet are examples of Transformer models, and comprise a subset of language models that have been shown to outperform other methods across various tasks, including sentiment analysis, question answering, and natural language inference (Wang et al., 2018).[3] While these models are considered to be a state-of-the-art in the natural language processing community, the tasks that these models are tested on are domain-agnostic which makes it difficult to assess feasibility of these models in a specific context. In this paper, we address this issue by focusing on political science text classification.

3.4.1 ULMFIT

The top panel of Figure 1 displays the architecture and working principle of the ULMFIT model (Howard and Ruder, 2018). The model is trained using a self-supervised objective based on predicting the next word in a sequence of words. The sequence is fed into the model, with each word represented as a vector similar to a word embedding vector. These vectors are then passed into a series of recurrent neutral networks, so called LSTM (long short-term memory) networks, that transform the input sequence into a single vector representation (Hochreiter and Schmidhuber, 1997). As opposed to using individual word embeddings, where the individual embeddings are averaged to obtain the representation of the sequence, here the output of the LSTM network at each step of the process is the representation of the sequence up to that point. This vector representation is then used to predict the most next word in the sequence and all parameters of the model are trained to optimize for this objective.[4] The intuition behind this and other similar language models is that if the vector representation of the sequence (or document) is successful at predicting what word should follow in the sequence, it likely encodes meaningful semantic and syntactical information that is contained in the input sequence. The

[3]At the time of writing this paper, RoBERTa and XLNet achieved the fifth and the sixth highest scores across a diverse set of existing natural language understanding (NLU) tasks, according to the General Language Understanding Evaluation (GLUE) leaderboard, which is a prominent evaluation framework for research towards general-purpose language understanding technologies (Wang et al., 2018).

[4] This part of the model is called the *decoder*.

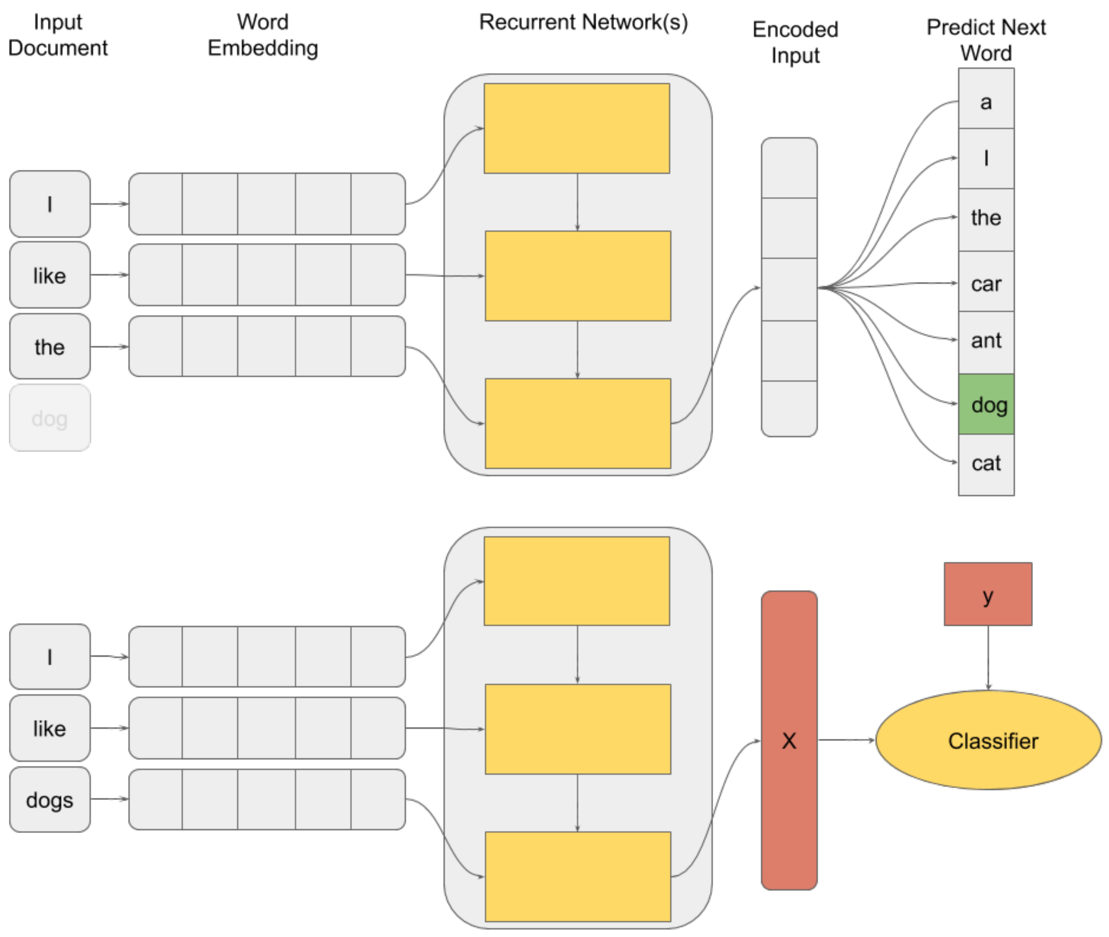

Figure 1: Illustration of the architecture of a deep language model. The top panel describes the logic of training language model, the bottom panel shows how the language model can then be used to train a classifier.

language model is thus trained on large corpora of language (specifically Wikipedia articles), similar to word embeddings, the training of the language model itself doesn't require labeled data.[5] After this the model can be used to encode any new or unseen sequence by simply passing it through the network. These representations then serve as a dense, low dimensional representation of the document that ideally encodes meaningful information that is relevant for a downstream task such as text classification. As seen in Figure 1 the final state vector so obtained contains all the context information for the entire sequence encoded within it to perform text classification in a manner similar to how a feature-set would be used by a more contemporary approach.

In order to use ULMFIT for supervised classification we follow the fine tuning procedure described in Howard and Ruder (2018): a model that has been pretrained on a large collection of text, such as the complete English Wikipedia database, is then fine tuned to the domain on which it is to be applied. This fine tuning allows the model to learn new words that are not in the vocabulary of the pretrained model (e.g. specific legislative terms in our application, hashtags in the case of social media posts) and adapt its parameters to the new domain. To perform text classification the obtained sequence representation is passed to a classifier, akin to how a feature set would be used, which is trained to to predict the label for the document as shown in Figure 1. This classifier is a simple 2-layer fully connected neural network (or any other classifier if desired) that maps from the k dimensional document representation to the m dimensions of the label space (i.e. the number of classes of the supervised problem).

3.4.2 Transformers: RoBERTa and XLNet

One of the main differences between ULMFIT and Transformer models is the fact that instead of LSTM units, Transformers use the multi-headed self-attention mechanism. This mechanism allows the model to gather information about the relevant context of a given word, and then encode that context in the vector that represents the word. This is achieved by allowing the neural network to weight all state vectors from the sequence and hence capture richer semantic

[5]By labeled data, we mean data labeled by human coders for the task in question (stance, relevance, or category).

information from the context. As discussed earlier, LSTMs use only the last state vector – and thus can capture only short-range dependencies – whereas Transformer models can use context about a word from distant parts of a sentence, which in turn helps the model to understand the word and its role in a sentence better. An attention function is illustrated in Equation 1. In this architecture, there are key (K), value (V), and search query (Q) vectors. The query searches over the keys of dimension d_k, or vector representations of all words, that might provide context for it. The meaning about the key word is encoded in the value vector. The output of the attention function is a weighted sum of the values, where the weight assigned to each value is determined by a dot-product of the query with all the keys (Vaswani et al., 2017).

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The attention function is performed in parallel on linear projections of queries, keys and values to d_k, d_k, and d_v dimensions, respectively, repeated h times producing d_v -dimensional output values. The final values are obtained by concatenating and projecting these output values once again.[6] This multi-head attention structure allows the model to capture different representations of the input, thus improving its accuracy (Vaswani et al., 2017).

In this paper, we implement two examples of Transformer models: RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019). The main difference between RoBERTa and XLNet is the training objective. RoBERTa is an example of autoencoding (AE), or masked language modeling, while XLNet is autoregressive (AR) model. In AE-based pretraining, given the input token sequence, a certain portion of tokens are masked and the model is trained to recover the original tokens using the context. XLNet is an example of AR language model, which predicts a future word given preceding context (Yang et al., 2019). Both models show extremely high levels of accuracy across natural language understanding tasks and thus can be considered as state-of-the-art models in transfer learning.

[6]For a more detailed explanation of the attention and multi-head attention mechanism please see Vaswani et al. (2017).

4 Data

We rely on three datasets to compare the different methods described in the previous section. First, we use data from both the bills and newspaper headline corpora of the Comparative Agendas Project Database (CAP). Second, we employ a collection of Tweets that are related to Hillary Clinton during the 2016 election campaign and are labeled for the tweet's stance on Clinton (positive, negative, neutral). Lastly, we use a corpus of Wikipedia discussion page comments that are labeled for containing hatespeech (yes/no). We use these four different types of text in an effort to evaluate the methods on a wide variety of datasets that could be potentially relevant to political science researchers.[7] These corpora vary in several substantive ways: they vary in their formality (bill headlines vs. tweets), their length (tweets and headlines are short, Wikipedia discussion comments are longer), and the dimensionality of the labeling task (the number of categories we label ranges from two – yes/no – to 21 policy categories for CAP data). We also construct the datasets in such a way that they span a range of sizes. As it can be seen from Table 1, the CAP data is very large (384,454 bill titles and 27,481 New York Times headlines), the Wikipedia hatespeech dataset is of medium size (42,768 comments) and the Twitter dataset is relatively small for a text classification task (2,450 tweets) .

Table 1: Datasets Description

Dataset	**Number of Documents**	**Labels**
Clinton tweets	2,450	Positive/ Negative/ Neutral
CAP Bills	384,454	21 categories
CAP NYT headlines	27,481	21 categories
Wikipedia comments	42,768	Yes/No

5 Analysis of Models' Performance

We conduct two types of analysis. In our first set of analyses, we test the performance of different models and representations on a 'classical' supervised task. That is, we use a random set of the labeled documents to train a classifier and evaluate it on a randomly held out

[7]See Appendix A.1 for more details on the datasets.

dataset; we will refer to this as the “train-test split" method. In the second type, we test the performance of these methods with a data availability constraint in order to evaluate how different methods perform at transferring information between tasks.

5.1 Tuning Parameters

For all the ‘bag-of-’ methods—that is, methods that do not account for the order of the words in the input documents—we use the following procedure to produce a reasonable baseline. First, using the training data obtained from the train-test split mentioned above, we use 5-fold cross-validation with 20 iterations of randomized search to obtain the best model over all tuning parameters.[8] Second, we test various combinations of classifier tuning parameters and feature representations. In particular, we test over the following ‘parameters’:

- **Classifier tuning parameters**: Algorithms used are Random Forest, Linear SVM, Logisitc regression with elastic net regularization
- **Gram size**: Word-ngrams: uni-grams, uni and bi-grams, uni, bi, and tri-grams; character n-grams: 3, 3 and 4, 3 and 4 and 5 grams
- **Pooling method**: Mean and max pooling (only used for embeddings)

The deep learning language model has several tuning parameters as well. Due to the large number of parameters and the large computational cost, we do a less comprehensive analysis in this case. For all language models, we optimize for the learning rate, which is the speed with which the model’s weights change and the number of epochs, or the number times that the learning algorithm will work through the entire training dataset. As a result of this optimization process, we obtain the set of optimal hyperparameters for each model. For ULMFIT, we also optimize for hidden size (number of features that are passed across the time steps of a samples when training the model), output size (number of outputs that are returned by particular LSTM layer), and dropout rate (which controls model regularization).

[8]Cross validation is a re-sampling procedure, where the data set is split into groups (5 in this case). Each group is used as a hold out or a test data set, while a model is being trained on the remaining groups. After the model is evaluated on the test set, the evaluation score is retained and the model is discarded.

5.2 Evaluation

After finding the best model with optimal tuning parameters by cross validating on the training data, we evaluate each model on the held out test set. For evaluation purposes, we use the proportion of correctly predicted labels (accuracy). We choose accuracy because the datasets chosen here are roughly balanced in their class distribution and accuracy provides an interpretable metric.[9] To estimate the variation of our results due to sampling variability in the test set, we bootstrap the accuracy metric by re-sampling the test set 50 times with replacement and re-evaluating the model on the resampled set.

5.3 Models' Performance on Traditional Classification Task

We investigate the performance of each method at extracting features (representing text) for downstream supervised learning on the datasets described above. For each of the feature sets – word n-grams, character n-grams, `fastText` embeddings, and `Word2Vec` embeddings – we evaluate three different machine learning classifiers: Linear SVM, Logistic Regression with elastic net regularization and random forest.[10] Additionally, we evaluate the performance of ULMFIT, RoBERTa, and XLNet models. We train and cross-validate the model on the training data and evaluate the performance of the best model from the cross-validation procedure on the held out test data.

5.4 Models' Performance on Classification Tasks with No Labelled Data From the Corpus to Be Classified

In the second analysis, we evaluate how well different supervised learning methods and text representations work in a harder classification task simulating a situation, where a researcher does not have sufficient labeled data for the task in question but where the labeled data for a related task is available (i.e., one has human labeled data available for the categories in which one is interested but from a different corpus). To do so, we make use of the fact

[9]See Appendix A.2 for F-1 score and classwise accuracy results for RoBERTa model

[10]For the datasets with more than two classes a one-vs-all classifier is trained for every label and these models are ensembled to provide a multi-class classification.

that the Comparative Agendas Project has labeled text from diverse corpora, including *New York Times* headlines and US Congressional bills, according to the same labeling schema. More specifically, observations in Comparative Agendas Project datasets are coded into one of 21 major topics. The examples of the topics include Macroeconomics, Civil Rights, Health, Economy, and Education.[11] We train a supervised model on the CAP Bills dataset and evaluate it on the CAP NYT data. In this case, the model has never 'seen' the NYT headlines when it produces predictions for them, as it was trained only on bill titles. We choose this task as it presents a realistic scenario for applied political science research: the CAP coding schema is a widely accepted way of categorizing political text into topic categories in political science. In doing so, we are essentially simulating a situation where a researcher does not have labels for the NYT data, which is the dataset of interest, but does have coded labels for the CAP bills (which could have easily been the case had the CAP project not chosen to include NYT headlines). Using the wealth of hand coded text that the CAP provides in order to apply it to other domains would extend the usefulness of the CAP corpus considerably, and be helpful for many potential political science applications.

6 Results

As described in the previous section, we begin first with our assessment of model performance on traditional classification tasks before moving on to the harder, cross-corpora, classification tasks.

6.1 Traditional Classification Task

Figure 2 displays the test accuracy of the supervised classification methods described above, across the four different datasets as described in Section 5. For each model and feature set combination we report only the best performing combination across all tuning parameters (for models as well as feature representations) according to the cross-validated accuracy. Results for each model are given in a different position along the X-axis, and the legend in the Figure

[11] We present a full list of CAP topics in Appendix A.1

indicates which feature set is being represented by the different color circles.

We find that character and word n-grams do equally well or comparable to word embeddings, but that language models (such as RoBERTa) consistently and robustly outperform the other schemes tested. Comparing the performance within each dataset, it is apparent that there is more variation between the various feature representations than between the classification algorithms used. Feeding the averaged or max-pooled embeddings features into a classical supervised algorithm is outperformed in all instances tested here by the simpler use of character and word n-grams as features. This finding suggests that bag-of-words methods are not only relatively simple and straightforward, but also an effective tool for text classification.

Regarding overall performance, both character n-grams and the language models performed consistently well across datasets and seem to be a safer bet in terms of a general *a-priori* recommendation for classification methods. It should be noted, however, that the character n-gram model needs to be properly tuned in order to achieve such high performance. For large datasets, this can mean a significant computational effort since the character n-gram feature matrices are usually quite large and thus the classification is likely to require large amount of memory. Models that are trainable with stochastic gradient descent (i.e. parameter updates can be performed with batches of data instead of the complete dataset), such as deep learning models, linear SVM and elastic net classifiers, require less memory than models such as random forest. To assist researchers in exploring these trade-offs – and implementing these different types of models – we provide an open source software package that allows for efficient cross validation of supervised models for text classification with little programming effort.[12]

While the absolute difference between models is relatively small (with the exception of the poorly performing linear SVM and logistic model with `fastText` and `Word2Vec` embeddings,[13]) RoBERTa – one of the Transformer models – significantly outperformed classical models in Clinton Tweets and NYT headlines datasets, which are the smallest datasets with the shortest data (and performed comparably or slightly better in the other two tasks). This is consistent with the argument made by Howard and Ruder (2018), that pretraining is most likely to

[12]The software is available on GitHub at *https://github.com/SMAPPNYU/smapp_text_classifier*
[13]These outliers illustrate the importance of cross-validation

improve the performance of the language model small amount of labelled data. Taking into account that RoBERTa is pretrained for the longest and on the largest amount of data (160GB of uncompressed text), the fact that it outperforms other models provides evidence for Howard and Ruder (2018)'s claim.

In order to check the robustness these results and verify that it is the size of the dataset and not the type of data itself that affects the difference in the models' performance, we compared the performance of RoBERTa to classical machine learning methods on the subset (10%) of CAP Bills dataset. Note that, in the first analysis, language models performed on par with bag-of-words and embeddings methods with random forest. As shown on Figure 3, RoBERTa achieved much higher accuracy on the sample of this dataset comparing to the other models.[14]

6.2 Classification Task with No Labelled Data From the Corpus to Be Classified

Figure 4 displays the results for our second analysis on the classification task with no labeled data from the corpus to be classified. Here the models are applied to *New York Times* articles without being trained on any *New York Times* articles. Instead the models are estimated on a labelled data set of Congressional Bills, then applied directly to *New York Times* articles (thus making the *New York Times* articles *truly* out of sample). In this scenario, language models outperformed traditional machine learning algorithms with the RoBERTa model achieving the highest accuracy score (61%). Given that we expect the language models to learn more abstract representations than bag-of-words or n-gram models, the better transfer learning properties make intuitive sense, and are in line with empirical results in other settings (Wang et al., 2018; Liu et al., 2019). These results not only provide evidence of the feasibility of using transfer learning models for political science tasks, but also suggest that these models can be extremely helpful in cases where the training data is absent or expensive to obtain but where labeled data from the relevant tasks is available. It is important to note, however, than while RoBERTa achieves the highest performance, it is still significantly lower than in the previous

[14]See Figure 4 in Appendix A.2 for the perfomance of RoBERTa across number of subset sizes

setting where RoBERTa was trained and tested on the same dataset (a decline in accuracy from 0.84 to 0.6). Thus, while the neural network-based language models perform better than other models in the setting when training data for the task in question in unavailable but the labeled data for the relevant task is present, having at least some amount of training data for the actual task appears to offer substantial improvements in accuracy.

Our findings therefore suggest that the Transformer models, and RoBERTa in particular, can help political scientists to classify text with high levels of accuracy and are likely to be particularly valuable – relative to more traditional methods of machine learning – when a given dataset is small and/or relatively little labeled data is available. While this conclusion is consistent with results from analyzing these models' performances on NLP tasks, our analyses both validate the use of, and illustrate the utility of, these models for label propagation classification tasks outside of the domain of NLP, and, in particular, for political science research. To assist researchers with implementing our recommendations, we also provide open source software for implementation of the version of RoBERTa model used in this paper.[15]

[15]The software is available on GitHub at *https://github.com/SMAPPNYU/SMaBERTa*.

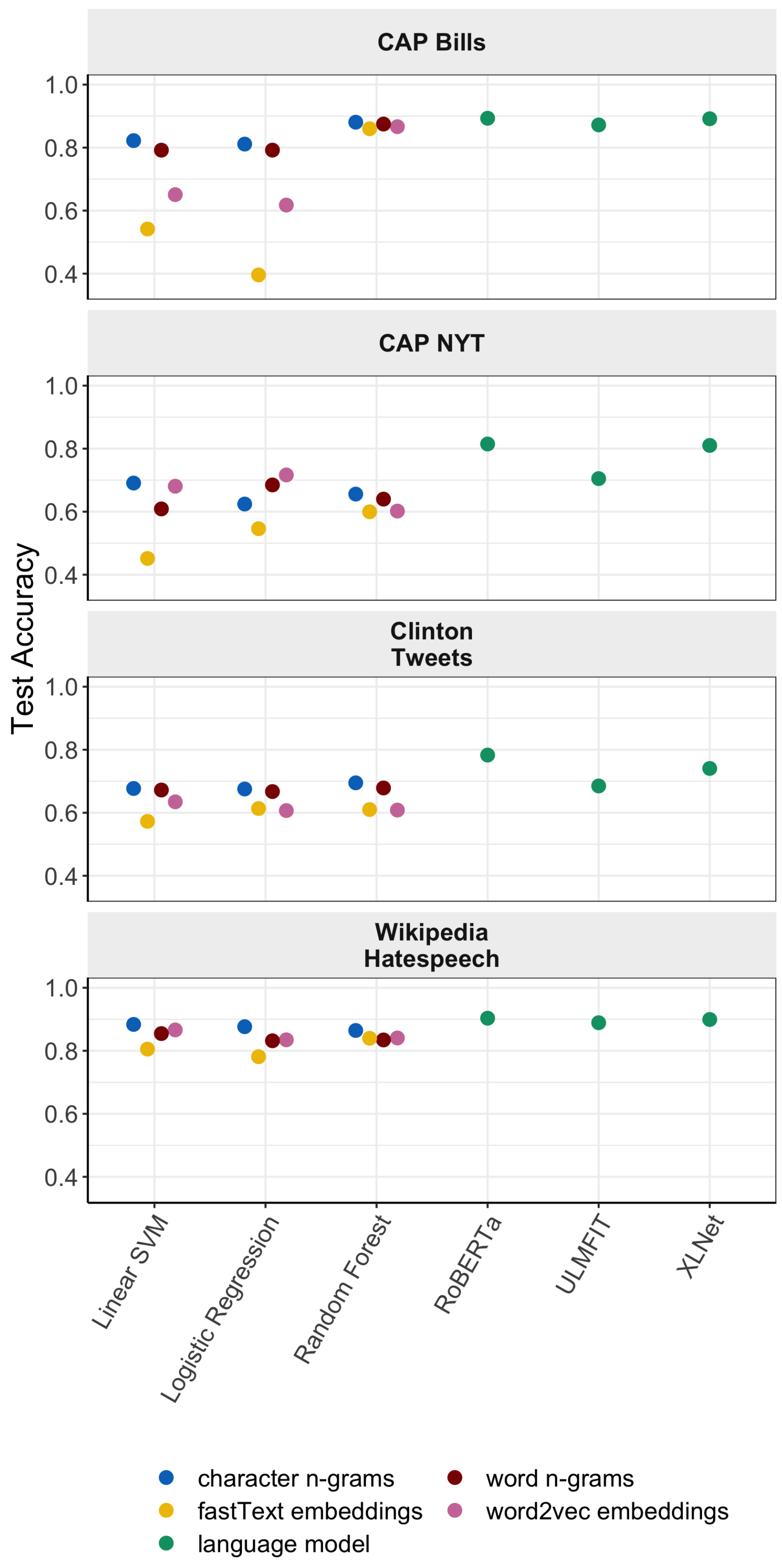

Figure 2: Results for traditional classification tasks across models and feature sets. Colors represent the feature sets. Test accuracy reflects the proportion of correctly predicted labels of the best model from the cross-validation procedure on the held out test data.

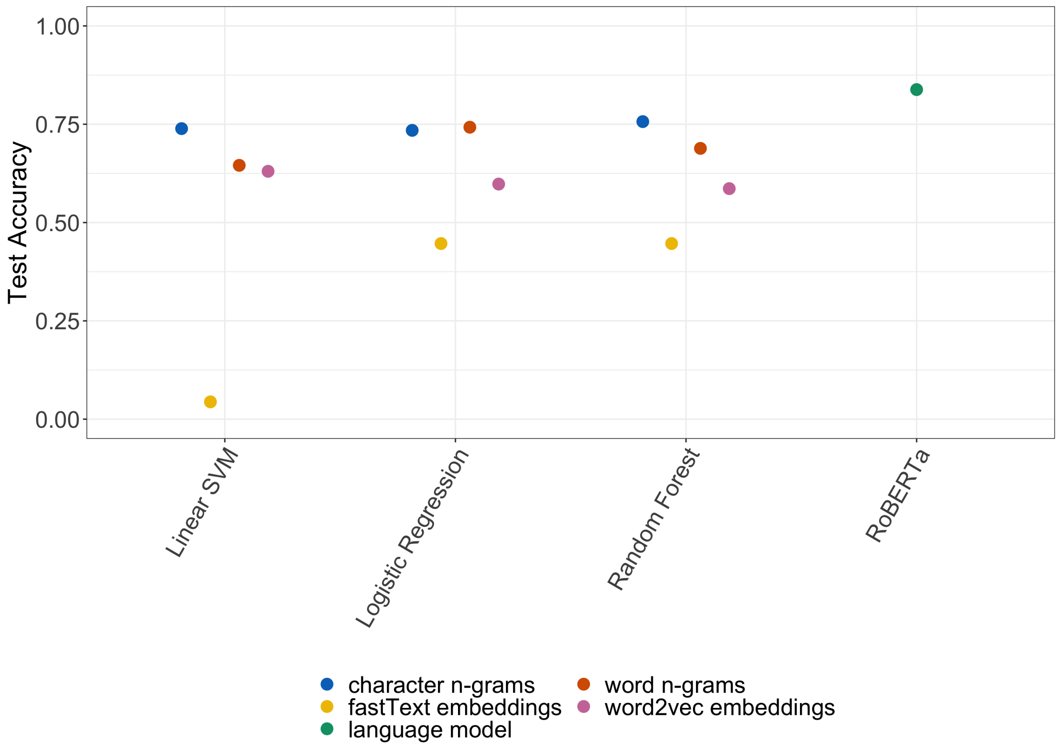

Figure 3: Results for the subset (10%) of CAP Bills dataset. The subset is a random sample of CAP Bills. Colors represent the feature sets. Test accuracy reflects the proportion of correctly predicted labels of the best model from the cross-validation procedure on the held out test data.

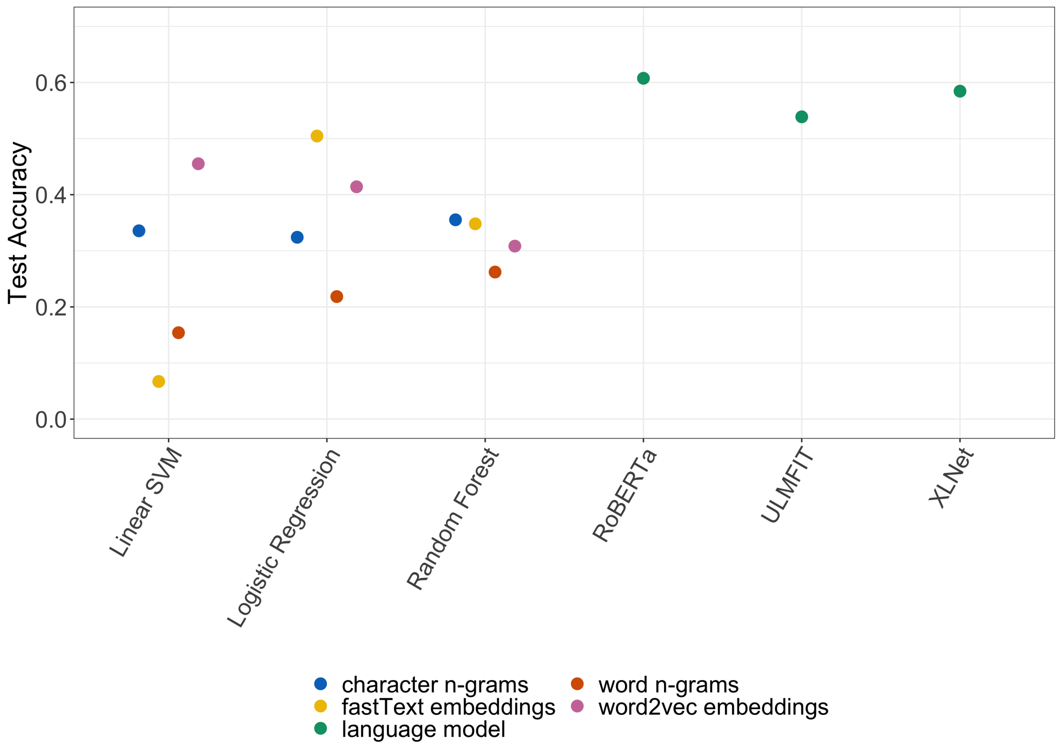

Figure 4: Results for classification task with no labelled data from the corpus to be classified. The models are trained on CAP Bills datasets and are evaluated on the CAP *NYT* data. Colors represent the feature sets. Test accuracy reflects the proportion of correctly predicted labels of the best model from the cross-validation procedure on the held out test data (CAP NYT dataset).

7 Discussion

We have discussed the most common methods for supervised text classification as well as some relatively new state-of-the-art transfer learning language models. We tested these approaches across various datasets in order to give applied social science researchers guidance in their choice among the large varieties of such approaches. In the direct comparison of performance at political science supervised text classification tasks, we found that language models performed consistently well across datasets, and Transformer based language models perfomed best out of the set of language models tested. Methods that used both pretrained and task-tuned language models for feature extraction generally performed better overall than classical methods, such as bag-of-words. Thus pretrained Transformer language models can provide political scientists with state-of-the-art accuracy while lowering computational and data labeling demands significantly.

When evaluating how well the methods perform when the domain of the training data is different from the domain in which the model was originally trained on (i.e., cases where no labelled data is available for the corpus to be classified), language models clearly outperformed the other methods. The highest levels of accuracy in such cases however are significantly lower than one are achieved by traditional methods, where a model is trained on the subset of the data set for which classification is required. Therefore, while as we demonstrated language models can potentially *reduce* the amount of labeled data necessary for achieving optimal levels of classification accuracy, we believe that our findings suggest that these models cannot *eliminate* the need for labeled data completely.

It is important to note, however, that while Transformer models show promise in getting classification results with relatively high levels of accuracy, they are likely to lose to more traditional machine learning models such as random forest in transparency and interpretability, because transformer models have many more parameters and a more complex architecture. We believe, however, that this complexity is warranted if human-level text interpretation and understanding is the goal, as it ultimately is in NLP and in application of NLP to political and social science data sets. By relying on architectures that are used (or can be adapted to) in a wide variety of settings and text-types, we hope to rely on the growing body of work aimed at interpreting these models, including literature that provides guidelines for pretraining, as well as literature that offers methods for interpreting the language models and their relationship to the text-feature space (Ghaeini et al., 2018; Guan et al., 2019).

Based on these findings we recommend that applied social science researchers consider using Transformer models, such as RoBERTa, when the amount of labeled data is relatively small. While Transformers are more likely to provide the best performance, we suggest that analysts always consider multiple possible methods when approaching supervised text classification tasks. For instance, in relatively easy tasks (ex. tasks where large training data is available), classical machine learning models are able to achieve relatively high levels of accuracy. In order to achieve these levels of accuracy, researchers should focus not only on the choice of a model, but also the feature set (corresponding feature extraction method) as well. The open-source software we are releasing in conjunction with this paper aims at making the implementation of such experiments easier and more computationally efficient.

References

Baumgartner, F. R., Green-Pedersen, C., and Jones, B. D. (2006). Comparative studies of policy agendas. *Journal of European Public Policy*, 13(7):959–974.

Chuang, J., Gupta, S., Manning, C., and Heer, J. (2013). Topic model diagnostics: Assessing domain relevance via topical alignment. In *International Conference on Machine Learning*, pages 612–620.

Denny, M. J. and Spirling, A. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, 26(2):168–189.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Firth, J. (1962). A synopsis of linguistic theory, 1930–1935; studies in linguistic analysis. *London: Philosophical Society*.

Ghaeini, R., Fern, X. Z., and Tadepalli, P. (2018). Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*.

Grimmer, J. and Stewart, B. M. (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297.

Guan, C., Wang, X., Zhang, Q., Chen, R., He, D., and Xie, X. (2019). Towards a deep and unified understanding of deep neural models in nlp. In *International Conference on Machine Learning*, pages 2454–2463.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lowe, W. and Benoit, K. (2013). Validating estimates of latent traits from textual data using human judgment as a benchmark. *Political analysis*, 21(3):298–313.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, Š., and Sedlmair, M. (2018). More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures*, 12(2-3):140–157.

Spirling, A. and Rodriguez, P. L. (2019). Word embeddings what works, what doesn't, and how to tell the difference for applied research. Working paper.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Volkens, A., Krause, W., Lehmann, P., Matthieß, T., Merz, N., Regel, S., and Weßels, B. (2018). The manifesto data collection. manifesto project (mrg/cmp/marpor). version 2018b.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.