DETECTION OF POLITICAL IDEOLOGY IN TEXT

A Comparative Study of Conventional NLP Techniques and Baseline Neural Network Models

Aditya Arcot Srinivasan

Nithyashri Govindarajan

Abstract:

Written or spoken textual content often give away the political ideology of a given person. In this project, we seek to determine the political ideology of a given sentence as opposed to longer speeches or corpora. In order to show the importance of the compositional aspect of language, we use a dataset that has been tagged with political ideology annotations at the sub-sentential level as well as the sentential level. Further, we seek to contrast the performance of traditional NLP techniques and language representation models like Bag-of-Words and wordlists against a Convolutional Neural Network (CNN). Based on our experiments, feature engineered traditional models perform on par with baseline neural network models.

Introduction:

Identifying political bias in a large portion of text like a speech is a difficult and time consuming process. Moreover, political bias is often fine grained and the political slant of a sentence is often derived from particular phrases used within a sentence. In this project, we aim to identify political ideology at a sentence level. In the context of US politics, we define a sentence to be politically biased if it gives cues as to the author's political position based on the words and phrases used in a given sentence.

In the scope of US politics and the following task, we define that a sentence is either liberal or conservative if it contains political bias that is indicative of 'liberal' or 'conservative' views as annotated by human understanding of the same.

In this project, we performed the following tasks::

1. Implemented classifiers using traditional Natural Language Processing (NLP) techniques, for example, Bag-of-Words (BoW) for Naive Bayes (NB) Classification and Logistic Regression.
2. Implement a basic Convolutional Neural Network (CNN) to detect ideological bias based on phrasal annotations at a sub-sentential level.
3. Compare the two approaches, check the applicability of both techniques and hypothesise as to which technique is better suited to discover compositional effects in text.

All work in this project closely shadows (Iyyer et al., 2014) and aims to hypothesise the relevance of traditional models and techniques with Deep Learning techniques for the correct detection of political ideology. One point of differentiation is that while political ideology detection is often performed at the speech or essay level, we seek to perform classify sentences at the

sentence level. This follows from semantic composition and making use of annotated political ideologies at the sub sentence level, which forms an important part of this classification task.

Related Work

Political Ideology detection has varied applications and thus justifies its growing interest. Much of the bulk of our project is based on the work done by (Iyyer et al., 2014) where they use sentential and sub-sentential annotations to detect political bias in text using Recursive Neural Networks that better captures the complex compositionality and polarity switches that a flexible phrasal annotation offers over a purely sentential annotation.

Much work has been concentrated on obtaining political bias at the document level and many techniques have been used in this process.

(Greene et al., 2009) examines identification of syntactic and semantic representations within text that indicate “implicit sentiment” of a sentence that is not visible superficially from the words alone. Combining lexical features with syntactic features, they obtain a relation between a pre-computed list of sentiment verbs and implicit sentiment.

(Wilson et al. 2005) describes the importance of phrase level polarities in order to effectively determine the overall sentiment of a given sentence. It presents a new approach to phrase-level sentiment analysis that first determines whether an expression is neutral or polar and then disambiguates the polarity of the polar expressions. This provides insights into why a given the ideology of a given sentence is the sum of the ideologies of its constituent phrases.

(Gentzkow et al., 2010) use a Bag-of-Words model to determine ideological leaning of newspapers to a Congressional Republican or Democrat by developing a “slant index” based on frequency of usage of certain discriminatory words. These are standard tokens that characterize the two political leanings. This is computed over multiple documents for every newspaper.

(Gerrish et al.,2011) used probabilistic models to determine politicians’ stances on specific issues as either right or left. They used Congressional Bills as a dataset. Within the view of supervised topics, the paper provides a link between the usage of language and the political view it represents, but this work is on a pure Bag-of-Words model.

(Sim et al., 2013) measured the ideological positioning of political candidates from their speeches. The authors built a system to infer ideological cues from a corpus of political writings previously annotated with known ideologies. Then speeches of US Presidential candidates are then represented as a sequence of cues and lags. A domain-based Bayesian Hidden Markov Model (CLIP) is then used to infer the proportions of ideologies every candidate uses in their speeches. The results are validated against a set of pre registered, domain expert authored hypotheses. However their model explicitly ignores intrasentential contextual influences that actually affect fine grained political bias.

(Biessmann et al., 2016) explores the idea of developing models to determine political leanings from a set of pre-election tweets, beyond basic sentiment analysis. . The methods make use of a variety of sophisticated NLP techniques in order to extract more meaningful and higher quality opinions, and incorporate extra-linguistic contextual information. In particular the authors leveraged bag-of-words feature vectors to train a multinomial logistic regression model. Cues were taken from the above model to implement our LR models.

Use of Deep Learning Techniques for NLP tasks is gathering a lot of steam with Recurrent Neural Networks often being described as most intuitive to understand the composition of text for any task. Since we use CNNs in our project, we look at related work using CNNs for sentiment analysis and sentence classification tasks.

(Lai et al., 2015) argues that usage of Recursive Neural Networks as in the case of the base paper we follow introduces unnecessary computational complexity. Semantic structure is captured in the form of a tree and its performance suffers in such a model of representation. Such a tree would take O(n^2) time in construction and usage where n is the number of words in a sentence, and thus unwise for longer sentences. Recurrent Neural Networks reduce the complexity to O(n) but are biased models that are heavily influenced by later words that it sees. CNNs however are unbiased models, but have filter restrictions in capturing semantic composition. Thus a combination of the latter two overcomes their respective weaknesses and maintains a respectable running time.

(Kim, 2014) uses CNNs for NLP tasks such as sentence classification particularly for sentiment analysis. The CNN using pre-trained word vectors from word2vec with little or no hyper parameter tuning produced very good results for this task, asserting that unsupervised pre-training of word vectors boosts performance greatly. We mirror this paper in the modelling and development of our CNN.

(Zhang et al., 2016) use CNNs of a single layer for sentence classification tasks because of their performance and reasonable comparisons to SVMs and Logistic Regression Baselines. They analyze the CNNs that are agnostic to heavy customizations and tuning that are rigorous and tedious to implement and offer guidelines for usage. Pre-trained word embeddings used to initialize weights increased the accuracy of the models but L2 regularizations do not contribute significantly to the accuracy. We use the latter factor in our implementation.

Dataset:

As stated above, and based on the work done by (Iyyer et al., 2014), the political bias in a sentence is based on sub sentential phrases and their ideological proportions in a sentence. For example, consider the following sentence:

“A wonkier version of the reform-equals-rationing argument is based less on panic mongering about Obama 's secret euthanasia schemes and more on the implications of something called `` comparative effectiveness research.”

Phrase labels for one liberal sentence phrase parse:

Liberal : A wonkier version of the reform-equals-rationing argument is based less on panic mongering about Obama 's secret euthanasia schemes and more on the implications of something called `` comparative effectiveness research .

Liberal : is based less on panic mongering about Obama 's secret euthanasia schemes and more on the implications of something called `` comparative effectiveness research

Liberal : based less on panic mongering about Obama 's secret euthanasia schemes and more on the implications of something called `` comparative effectiveness research

Neutral : the implications of something called `` comparative effectiveness research

Neutral : something called `` comparative effectiveness research

Neutral : called `` comparative effectiveness research

Neutral : comparative effectiveness research

The above parse for of a given sentence has the sentence broken into phrases with the annotation for each of the phrases. The smaller neutral phases when combined change polarity to Liberal when viewed as a whole. This reinstates compositionality.

Phrase labels for one conservative sentence phrase parse:

“In response to ROBERT FRANK 'S cover story , which argues that consuming less can make our society better , WONKETTE writes , `` Maybe we won't ever have indulgent capitalist luxuries like homes or jobs anymore , like we did in 2007 , but maybe that 's cool now.”

Conservative : In response to ROBERT FRANK 'S cover story , which argues that consuming less can make our society better , WONKETTE writes , `` Maybe we won't ever have indulgent capitalist luxuries like homes or jobs anymore , like we did in 2007 , but maybe that 's cool now .
Liberal : won't ever have indulgent capitalist luxuries like homes or jobs anymore , like we did in 2007 , but maybe that 's cool now

As seen in the above sentence, the liberal phrase would not be classified as a conservative sentence without being taken as part of the bigger context.

The different phrases and punctuation cause the sentence to switch between liberal and conservative labels when looking at the sentence as a whole.

Along these lines, the work done by (Iyyer et al., 2014), involved the creation of an annotated subset of a corpus called the Ideological Book Corpus (IBC) developed by (Gross et al). The IBC is a collection of books and magazine articles between 2008 and 2012 by authors with known political leanings. Every document in the IBC has been labelled with ideologies (left, right and center). Therefore, the IBC was filtered and annotated using the CrowdFlower crowdsourcing platform to obtain human annotations of the filtered IBC dataset for political bias at the sentence as well as phrase level.

We obtained the filtered annotated subset of the IBC from the first author of the paper we are shadowing, Mohit Iyyer. This dataset will form the basis of our work.

The dataset consists of 4,062 sentences annotated for political ideology at a sub-sentential level.
There are three types of tags overall:

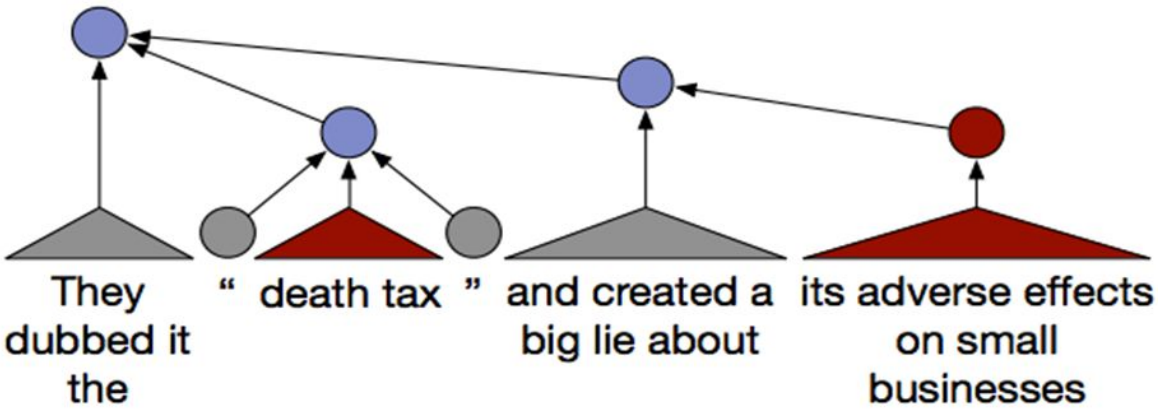
1. Liberal
2. Conservative
3. Neutral

Specifically, it contains:

1. 2025 liberal sentences,
2. 1701 conservative sentences, and
3. 600 neutral sentences.

Each sentence is represented by a parse tree where annotated nodes are associated with a label in {liberal, conservative, neutral}.
An example is as follows:

Liberal Sentence	Conservative Sentence	Neutral Sentence
Forcing middle-class workers to bear a greater share of the cost of government weakens their support for needed investments and stirs resentment toward those who depend on public services the most .	Gore is getting rich from environmentalism , not just by being paid a whopping $ 175,000 per speech but by using political pressure to force government policy in a direction that benefits his business interests .	In this country , the beneficiaries of Apple 's success are , first , the designers , who have done wonders working with Steve Jobs to produce products that are beautiful and effective .



The above figure is an example of compositionality in ideological bias detection (red → conservative, blue → liberal, gray → neutral) in which modifier phrases and punctuation cause polarity switches at higher levels of the parse tree.

Some additional statistics about the dataset:
The following represent the the top uni-grams for each category, with their counts.
We see that a lot of frequently occurring words are common across the two categories, but there are certain discriminatory terms that are clearly characteristic of only one class. We can validate that with human annotation - liberal sentences have words like “tax”, “system”, “workers” whereas conservative sentences have words like “free”, “american”, “freedom”, both categories of words are more likely to occur with those labels only.

Liberal	Frequency Count	Conservative	Frequency Count
would	235.0	government	321.0
tax	228.0	would	190.0
government	199.0	people	171.0
people	192.0	free	158.0
care	192.0	economic	140.0
health	189.0	one	135.0
public	175.0	make	133.0
energy	171.0	federal	122.0
new	159.0	money	109.0
economic	157.0	new	104.0
make	151.0	freedom	102.0
social	151.0	market	101.0
system	131.0	economy	101.0
workers	128.0	american	99.0
one	123.0	energy	99.0
create	120.0	state	99.0
rich	119.0	care	98.0
even	117.0	social	95.0
cut	117.0	even	94.0
economy	116.0	create	94.0
low	115.0	created	88.0
free	114.0	health	88.0
income	113.0	since	86.0
class	113.0	party	86.0
security	111.0	use	84.0
use	110.0	control	83.0
also	107.0	jobs	81.0
market	107.0	many	80.0
good	106.0	also	77.0
well	102.0	could	77.0
jobs	100.0	well	77.0
political	98.0	system	76.0
party	96.0	like	76.0
like	95.0	public	75.0
many	93.0	political	74.0
poor	93.0	security	72.0

The IBC data is an annotated data set, where every sentence is labelled into one of three categories - conservative, liberal and neutral. The dataset is distributed only as an annotated dataset by the authors of (Iyyer et al., 2014) through mechanisms described in the previous section. Further, the IBC has annotations at the phrasal level as well and this was the state in which the dataset was acquired by us. Therefore, human annotation was not required with the IBC Corpus.

The full balanced IBC dataset consists of 3,412 sentences (4,062 before balancing and removing neutral sentences) with a total of 13,640 annotated nodes, where a node represents a tree formed by parsing the sentence using the Stanford Parser. The IBC dataset does not contain more than one tree for every sentence (corresponding to multiple paths), instead keeps only path and the labelling is decided based on this path. The paths are selected based on the ones that have the most partisan unigrams, quoted as "The words that the multinomial Naive Bayes classifier in *DUALIST* (DUALIST is a utility for learning semantic terms for NLP tasks) marked as highest probability given a polarity: market, abortion, economy, rich, liberal, tea, economic, taxes, gun, abortion", if none such then the ones that have the most open classes of parts-of-speech. The nodes are annotated bottom-up to prevent the user from changing lower levels of annotation based on the root level annotation.

A note about the IBC dataset: the lower level phrases almost always were "neutral" while a compositional set of phrases or full sentences were tagged with a non-neutral label. Thus, the neutral sentences are removed from the IBC to enable the classifiers to better learn the compositional structures that contribute to the bias of a sentence as opposed to just being influenced by the lower nodes.

Methods and Experiments

The task involves three concurrent objectives:

1. Classification using traditional Natural Language Processing (NLP) techniques, for example, Bag-of-Words (BoW) for Naive Bayes (NB) Classification and Logistic Regression (LR).
2. Classification using a neural network to detect ideological bias based on phrasal annotations at a sub-sentential level.
3. Comparison of the performance of the two approaches

Conventional Models:

We run a few baseline algorithms to establish the range of accuracies we can expect.

Like (Iyyer et al., 2014) we do not include neutral sentences in our task because we do not want the lower level phrases which tend to be mostly neutral to affect the features learnt by our classifier, especially in case of longer sentences where intuition drives that the sheer number of such neutral phrases could influence the features learnt by the classifier or the compositional structure of the sentence that could lead to inaccurate predictions.

Baselines:

There are three baselines, and all are run on sentence level features.

Random:

The Random Baseline picks at random a class for each sentence - one of Liberal or Conservative. Trivially, the accuracy of the Random Baseline is expected to be 50%.

Most Common Label:

We examine the dataset in our training data and assign to each iteration of the test sample, the label of the most commonly occurring class.

Naive Bayes Classifier Using a Bag-of-Words Model:

Naive Bayes assumes the independence of the features. Since we are using Bag-of-Words implementation, or a unigram model, the features are just the words and their associated counts. We do not perform any stemming/removal of stopwords for this classifier and use phrases as a whole. This algorithm is executed for 10-cross validations, with a test-train split of 90%-10%.

As additional baselines, Naive Bayes is run on bigram and trigram models as well.

Experiments:

Logistic Regression:

Logistic Regression is used as a classifier. Starting with a baseline Logistic Regression with just Bag-of-Words features at the sentence level, then add additional features at both the sentential and sub-sentential levels. All runs are executed for 10-cross validations, with a test-train split of 90%-10%.

Logistic Regression Using a Bag-of-Words Model (LR-1):

This is the most basic form of logistic regression for this dataset, using only BoW features in the model. Running a logistic regression and a Naive Bayes classifier using the same features and comparing their accuracies will help draw some conclusions about the nature of the dataset and the applicability of these classifiers to this dataset.

For comparisons with the Naive Bayes baselines, we run this on bigram and trigram models of the sentence words.

Logistic Regression with Sentences and Phrases as Bag-of-Words (LR-2):

LR-1 model with phrasal annotations as more training data. This has both the sentential words and their counts, and annotated phrases as training data.

Logistic Regression Model on Sentences with Word2Vec embeddings (LR-w2v):

Logistic regression over averaged word vectors of the sentences with pre-trained word embeddings from word2vec. The word embeddings are obtained by the word2vec toolkit vectors of 300 dimensions trained on around a 100 billion words from the Google News corpus (Mikolov et al., 2013). This provides a baseline that encodes both semantic and syntactic information later used for comparison with the Neural Network Model.

Logistic Regression Model on Sentences with Word2Vec embeddings (LR-w2v-2):

Lr-w2v with averaged word vectors of both sentences and annotated phrases, using word2vec toolkit vectors trained on the Google News Corpus.

Neural Networks:

A Convolutional Neural Network (CNN) based on the architecture of (Kim, 2015) and modelled as per (Britz, 2015) is used for the classification task. A CNN is a feedforward network that has several layers of “convolutions” or a filter run over the input to produce outputs. So instead of traditional feedforward neural networks, a convolution produces output fed to the subsequent layers. This is capped off with pooling and softmax layers that subsample inputs and reduce outputs to succinctly represent the output. During the training phase, the CNN learns values of its filters used in the convolution based on the task. Further, with every word represented as a vector, word embeddings can be used to initialize the weights in the classification task.

Convoluted Neural Network with random initialization of weights (CNN):

CNN with word embeddings learnt from scratch (weights initialized to 0) at the sentence level without any L2-regularization - a method to overcome overfitting in the training phrase by adding the error function to the square of magnitudes of all weights in the network for that run. This is run for a 90%-10% split in train-test data.

This is implemented in TensorFlow as per (Britz, 2015). The CNN with the convolution identifies semantic compositionality and structure in the input text.

As a comparison, we run the model with the same setup but L2-regularization of 0.15.

In this model, it is to be noted that we intend to only establish a baseline for Deep Learning techniques to be used in comparisons to traditional models. Rigorous feature engineering is not part of the scope of this project.

Results:

All the experiments conducted were performed on a 90%-10% training test split along with 10-fold cross validation.

The Naive Bayes and Logistic Regression models were run with training instances incorporating 1 gram, 2 gram, 3 gram and (1-5) gram as illustrated in the figure below.

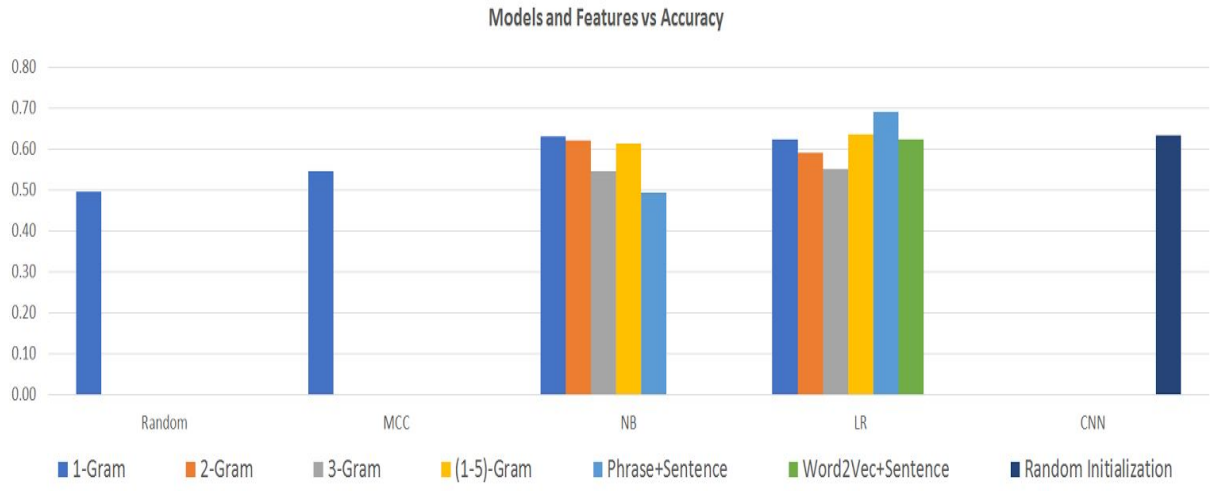
In addition, both the Naive Bayes and Logistic Regression models were run with only sentences and (sentences + phrases) as training instances.

The LR model was also run incorporating word embeddings from the Google News Dataset; every sentence becomes a feature vector representing the average of the word embeddings for a given sentence. This was not carried out with (sentences + phrases) because of time and memory constraints.

Lastly, we put forward a baseline CNN implementation which used only sentences as training instances, with a random initialization of initial weights to the model.

The figure below illustrates the above information succinctly:

	1-Gram	2-Gram	3-Gram	(1-5)-Gram	Phrase+Sentence	Word2Vec+Sentence	Random Initialization
Random	0.495978552	-	-	-	-	-	-
MCC	0.545472636	-	-	-	-	-	-
NB	0.631499274	0.62022537	0.54710334	0.61395788	0.493611525	-	-
LR	0.622829415	0.59041323	0.55212368	0.63709526	0.690146983	0.623213233	-
CNN	-	-	-	-	-	-	0.632708



As seen above, the highest accuracy obtained was from the the Logistic Regression Model which used (sentences + phrases) as training instances.

CNN without L2-Regularization has an accuracy of 63% whereas with L2-Regularization of 0.15 has an accuracy of 58%.

Analysis:

We analyze our results at multiple levels, we take a look at the different models grouped under conventional techniques and analyze their performance based on the task.

Baseline Algorithms:

The baselines are trivially explained. The Random classifier assigns a label by randomly picking uniformly, thus achieving an accuracy of close to 50%, which is expected. The Most Common Classifier assigns “Liberal” to every sentence, since Liberal sentences outnumber Conservative sentences 2025-1701 for a 54% accuracy.

Naive Bayes vs Logistic Regression:

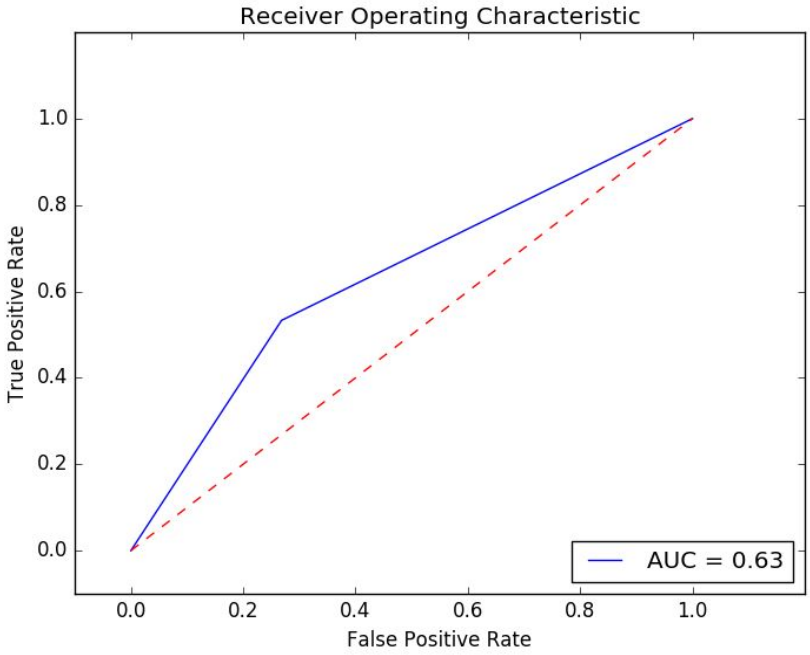
The Naive Bayes model and Logistic Regression models are run on multiple features. We analyze their performance by a Receiver Operator Characteristic Curve.

Receiver Operating Characteristic:

A receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

We have plotted two ROC curves, one for the Naive Bayes Classifier and the other for the Logistic Regression classifier for the unigram models at the sentence level of annotated data.

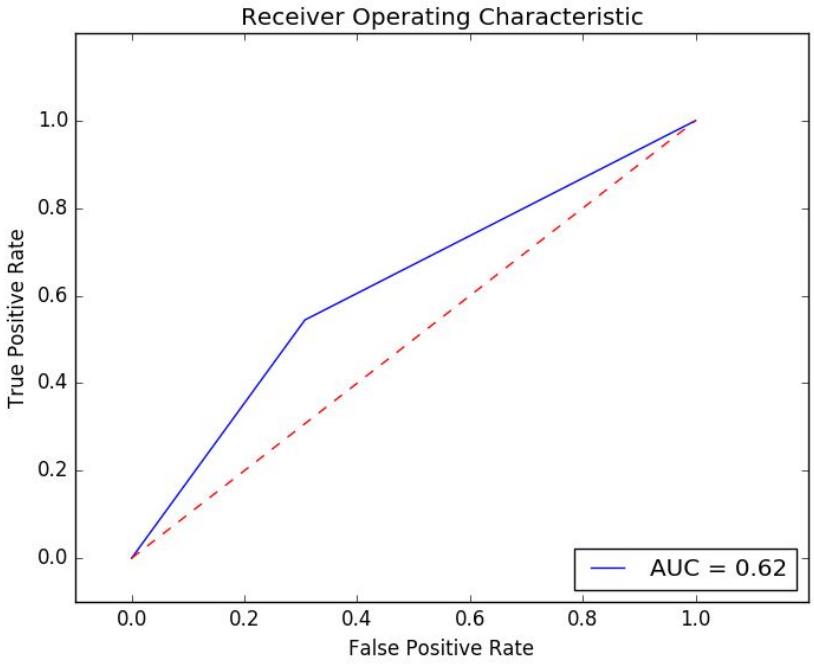
AUC stands for “Area Under the Curve” : the area under the curve is equal to the probability that a classifier will rank a randomly chosen positive instance(liberal) higher than a randomly chosen negative one(conservative) (assuming 'liberal' ranks higher than 'positive').



The ROC curve for Naive Bayes algorithm, produces an area under the curve of 0.63

The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

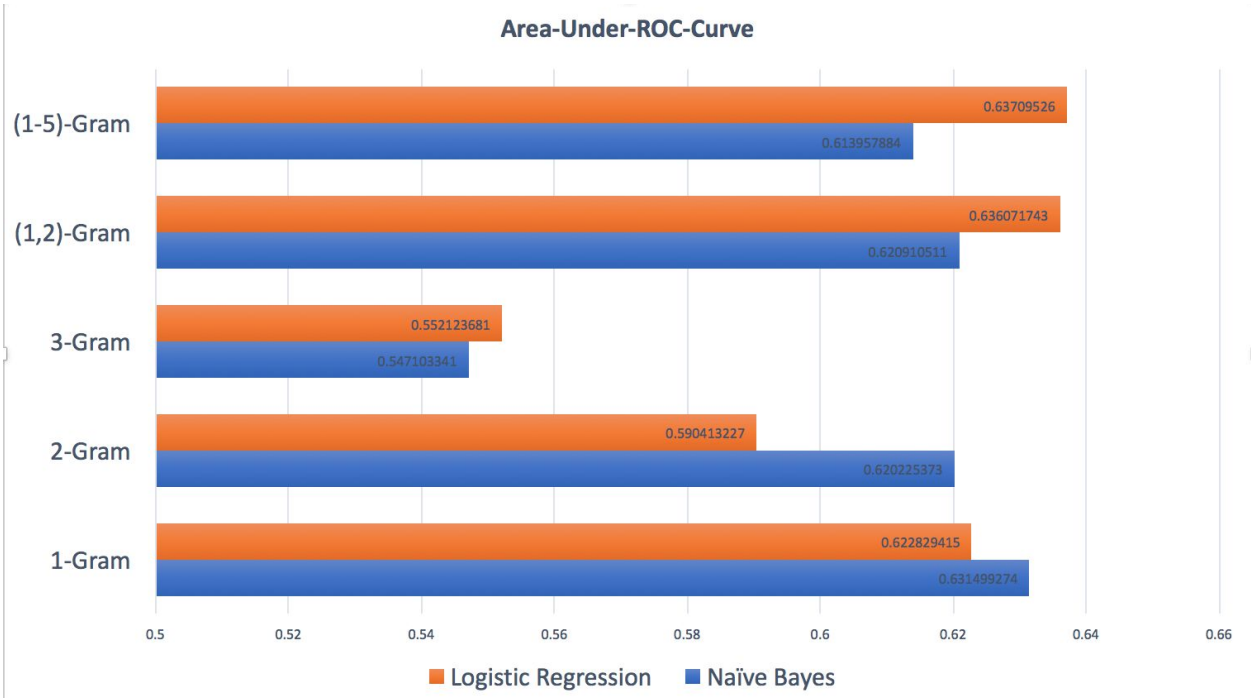
A perfect ROC curve would have an AUC of 1.0



The ROC curve for LR-1 algorithm, produces an area under the curve of 0.62

Naive Bayes has a better accuracy than LR-1, by about 1%. We hypothesize that this is because of the scarce training samples. A quick reading of related literature tells us that a Naive Bayes model, with its feature independence assumption, converges to its asymptotic values in log n examples whereas a Logistic Regression model, which usually performs better with more training samples, converges to its asymptotic values in n examples. Here we have a relatively small training sample of 2025 liberal sentences and 1701 conservative sentences and thus, the Naive Bayes algorithm performs better.

We hypothesise that the ROC curves for both classifiers have a “sharp” bend because of the relative scarcity of data points.



Further, by examining the AUC for Naive Bayes and Logistic Regression for different models on sentence data, as represented in the graph above, we see that Logistic Regression outperforms Naive Bayes from the trigram model onwards, when the feature independence of trigrams of sentences as features does not take into account the compositionality and the semantic structure, thus affecting the classification task.

With (1,2)-Gram and (1,5)-Gram representations, the number of data points in training helps the Logistic Regression models, posting its highest accuracy of 63.6% and 63.7% respectively.

LR-2 models posts the best accuracy of all our experiments, with a 69%, the Naive Bayes Model with the same features posts the least accuracy of all Naive Bayes runs, with a 49%. This is explained by the exponential increase in the number of phrases per sentence, thus the training samples blow up. The LR-2 model captures the compositionality of text from these phrases and is able to learn that, the phrasal annotations help capture this. With Naive Bayes, the feature independence ignores this obvious structure in sentences where compositionality of words expresses political inclinations, and thus based on just independent features of data, the accuracy is really poor.

Logistic Regression:

Analyzing the different features for logistic regression models, LR-2 posts the best accuracy with 69%. This model incorporates both sentence and phrasal features. This model is able to detect the semantic compositionality of text - in discriminating the features that classify the sentences into liberal or conservative. The discriminatory features are phrasal annotations, and this helps the classifier learn even the semantic structure text.

LR-w2v uses sentence features trained on word embeddings using the word2vec toolkit. It has an accuracy lower than LR-2. This brings to light the importance of the phrasal annotations and semantic compositionality of text. Even with pre-trained word embeddings that capture word ordering that a Bag-of-Words representation ignores, the sentence features alone do not help increase the efficiency of the model. LR-2 has close to a 7% higher accuracy. This implies that the semantic compositionality in text have a higher importance in predicting political leanings inherent in the text rather than just order of words.

A note on Lr-w2v-2 where average word embeddings are used alongside sentence and phrasal features is that the model is computationally expensive and memory intensive (blown up to over 1 million data points). Our limited resources were unable to complete the task in entirety and any form of reduced representations would be unfair comparisons to the other runs where complete representations of both sentence and phrasal features are used. This task is part of the future work.

Logistic Regression vs CNN:

The CNN, as per our literature survey, while traditionally used for image classification task, performs well with NLP tasks because of location invariance and local compositionality that is captured by the convolution. The structure and semantic undertones of a phrase of text, and further a sentence is captured by the weights. Our baseline implementation of the CNN without word embeddings used to initialize weights and on only sentence features saw an accuracy of 63%. This is much lower than LR-2. The inclusion of the phrasal features seem to outweigh the semantic compositionality captured by the neural network on just words of the sentences.

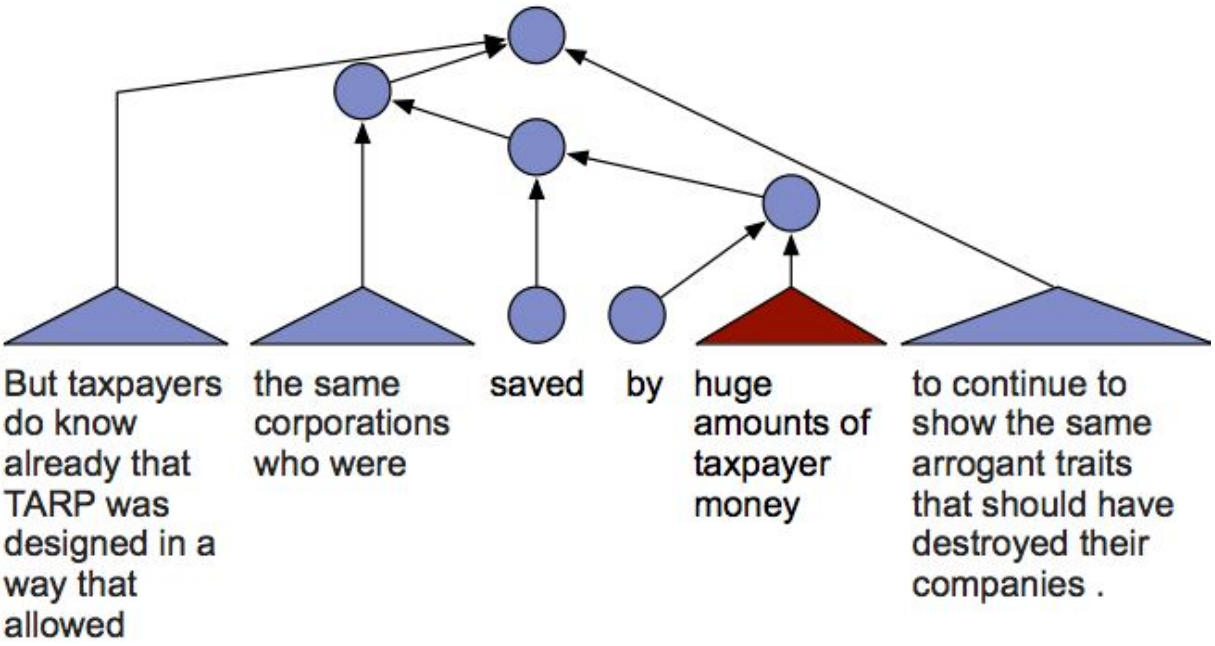
Discussion and Future Work:

In our comparison between traditional NLP techniques versus a Convolutional Neural Network (CNN), the LR-2 model with (sentence + phrases) as training instances came out on top. In fact, from (Iyyer, et al, 2015) we see the following results for experiments on the IBC dataset for a 10-fold cross validation for sentence level political bias detection.

Model	Convote	IBC
RANDOM	50%	50%
LR1	64.7%	62.1%
LR2	–	61.9%
LR3	66.9%	62.6%
LR-(W2V)	66.6%	63.7%
RNN1	69.4%	66.2%
RNN1-(W2V)	**70.2%**	67.1%
RNN2-(W2V)	–	**69.3%**

While the neural network models used here are Recursive Neural Network models, which are different from the baseline CNN (features corresponding to RNN1) we have used, the accuracy still is lower compared to the LR-2 model (not to be confused with the LR2 model in the above table) has an accuracy comparable to the RNN2-(w2v) model that has sentence and phrasal features initialized with average word2vec word embeddings as weights - the same features as our LR-w2v model, yet the accuracy of RNN2-(w2v) is comparable to our LR-2 model.

LR-2's performance can be attributed to the features used in it. Phrasal features are the most important features for this classification task, because they help any classifier get better insights about the semantic compositionality of text, where in any sentence, certain words or certain combination of words may represent one political ideology whereas in combination with other words may come to represent something else altogether. This compositionality is crucial in this task, because the classification is not performed at a word level, but a sentence level as a whole. As a composition of multiple phrases, the political ideology of a sentence can truly be captured by the nature of the semantic composition of these phrases. Further, polarity switches in phrases represent a complex problem which can be handled only by phrasal features.



Polarity switches and phrasal level semantic composition of a sentence - a combination of phrases results in different labels, different phrases in a sentence also have differing labels. Implicit bias in a sentence is leveraged by the phrasal composition and semantic structure, not visible to directly obtainable lexical features.

We believe that using the LR model with pre trained word embeddings for phrases along with sentences would perform better, as is the case with the CNN. However as per the scope of this project, we hypothesise that a well engineered traditional model will outperform or perform as good as a baseline neural net model while being far easier to implement. In cases where the ease of implementation of the models is crucial and features of text are easily available and mouldable as input to the model, a conventional discriminatory model with additional features

performs better than neural nets that require rigorous training and constant hyper parameter tunings over multiple iterations.

We intend to explore this area beyond the scope of this course to more closely investigate how the above models can be tweaked for better performance. The immediate next steps would involve:

a) LR model with pre trained word embeddings for (sentences + phrases)
b) CNN model with (sentences + phrases)
c) CNN model with pre trained word embeddings (sentences + phrases)

One obvious area of future work would include detection of fine grained ideology detection beyond just liberal or conservative. However, a suitable dataset would have to be acquired for this task or perhaps the detection of latent political ideologies as across blogs or social media. Finally, while our task is restricted to sentence level classification, combining this with document level or speech level classification might yield better results at all levels.

Acknowledgments:

We thank our project mentor Abram Handler, for his insightful comments and feedback. We would also like to thank our Professor, Brendan O'Connor for the opportunity to take on this class project and Mohit Iyyer, the first author of the base paper we closely shadow in our project for his help with the dataset, quick comments on the techniques and clarifications with our results and understanding.

References:

Felix Biessmann, Pola Lehmann, Daniel Kirsch, Sebastian Schelter. 2016. Predicting Political Party Affiliation from Text. *International Conference on the Advances in Computational Analysis of Political Text (PolText).*

Matthew Gentzkow and Jesse M. Shapiro. 2010. What drives media slant? evidence from u.s. daily newspapers. *Econometrica*, 78(1):35–71.

Sean M. Gerrish and David M. Blei. 2011. Predicting legislative roll calls from text. In *Proceedings of the 28th international conference on Machine learning*, ICML ’11.

Stephan Greene and Philip Resnik. 2009. More than words: syntactic packaging and implicit sentiment. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL ’09, pages 503–511, Stroudsburg, PA, USA. Association for Computational Linguistics.

Justin Gross, Brice Acree, Yanchuan Sim, and Noah A Smith. 2013. Testing the etch-a-sketch hypothesis: A computational analysis of mitt romney’s ideological makeover during the 2012 primary vs. general elections. *APSA 2013* Annual Meeting Paper.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, Philip Resnik. Political Ideology Detection Using Recursive Neural Networks. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL pages 1113–1122, Baltimore, Maryland, USA, June 23-25 2014.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882.*

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. *Twenty-Ninth AAAI Conference on Artificial Intelligence.*

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*

Yanchuan Sim, Brice Acree, Justin H Gross, and Noah A Smith. 2013. Measuring ideological proportions in political speeches. *Conference on Empirical Methods in Natural Language Processing*.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820.*

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pp. 347–354.*

CNN implementation guide and code: https://github.com/dennybritz/cnn-text-classification-tf