Determining Support/Opposition using a Recursive Neural Network

Sarah Widder

Abstract

Evaluation of persuasive political documents is extremely important in the current political climate. In this paper, I attempt to apply recursive neural network methods to the task of determining whether a speaker supports or opposes a certain piece of legislation, using publicly-available voting records as ground truth. Using Convote, a dataset of transcripts from 2005 debates in the U.S. House of Representatives, the network was able to reach 65% classification accuracy when run on about 600 parse trees discussing H.R. 5. Possible reasons for the network's inaccuracy and additional methods for improving performance are also discussed.

1 Introduction

As noted by Thomas et al. (2006), we are increasingly surrounded by politically oriented text, especially online. This prevalence of politically oriented texts has become even more important in recent years. Official government documents, transcripts, laws, and proceedings are available online, and with the growth of social media, debates formerly confined to Congressional committees and conference rooms are now beginning to take place publicly online. Campaigns for political office include massive amounts of information and advertisement online intended to convince voters of the merits of a candidate and his or her policies. As we also grapple with determining the authenticity of news and increasingly ideologically opposed parties, it is apparent that we need better ways to process and evaluate persuasive documents to determine both ideological bias (Iyyer et al. 2014) and support vs. opposition for a cause or piece of legislation (Thomas et al. 2006).

In this paper, I attempt to apply recursive neural network methods to the support vs. opposition task first approached by Thomas et al. (2006). I seek to determine from the transcripts of U.S. Congressional floor debates (the Convote dataset) whether each one-sentence utterance represents support for or opposition to a proposed piece of legislation, using the publicly-available record of the speaker's vote on the legislation as ground truth. The recursive neural architecture is intended to incorporate the syntactic structure of speech segments, allowing the syntactic relations between relevant clauses to be used when determining the sentiment of the overall sentence. In particular, I am interested in whether a recursive neural network can learn and model the syntactic and semantic relations that are involved in persuasive speech. This is an important linguistic skill in humans and computational models.

2 Related Work

Thomas et al. first (2006) created the Convote dataset and attempted the task of support/opposition classification. Their approach focused heavily on first determining agreement between different speakers, and then using this to aid a support vector machine classifier in determining the support or opposition sentiment of the speaker. Agreement classification involved identifying other Congressmembers referenced in the speech segment and determining whether the speaker agreed or disagreed with them. Speech segments were represented using plain unigrams as features, input to the SVM as normalized presence-of-feature vectors. They achieved 70.81% accuracy on document-level classification using an SVM combined with links indicating speeches by the same speaker and agreement with other speakers.

The determination of support vs. opposition is a specific application of the wider task of sentiment analysis, a very important field of natural language processing. A recursive model for sentiment analysis was first proposed by Socher et al. In their 2013 paper, they introduce the Stanford Sentiment Treebank of fully labeled parse trees corresponding to sentences from movie reviews. They then apply a Recursive Neural Tensor Network to these trees, which is able to accurately predict the compositional semantic effects present in the sentiment treebank.

While the Convote dataset created by Thomas et al. has seen wide use in other NLP research (see http://www.cs.cornell.edu/home/llee/data/convote.html), it has only recently been used for sentiment analysis using neural or deep learning techniques. Iyyer et al. (2014) present a recursive neural network model for political ideology detection, using a Convote dataset filtered for ideologically biased language and politically "sticky bigrams" (such as "death tax", "corporate greed", etc.). Their recursive neural network model using word2vec embeddings achieved 70.2% accuracy in predicting the party (and thus the assumed liberal/conservative bias) at the sentence level for Convote speech segments.

While semantic word embeddings and neural architecture was used on the Convote dataset by Iyyer et al., the original task of determining support vs. opposition has not been attempted with these new approaches. In this paper I attempt to train a recursive neural network on parse trees from the Convote dataset with the aim of predicting the speakers eventual vote on, and thus their support of, the legislation under discussion. My intent is for semantically and syntactically relevant linguistic aspects of the speech segments to come out through the recursive neural architecture, and that this will offer an improvement over the statistical methods first applied to this task by Thomas et al. in their original paper.

3 RNN Approach

For this task, I used a simple recursive neural network modeled from one used in a homework assignment from Natural Language Processing (CPSC477). The input to the network is a syntactic/dependency tree (produced by the Stanford Parser, discussed later). At the leaf nodes, the activation is the embedding of the word at that node. For non-leaf nodes, the activation takes the sum of the activations of its child nodes, multiplies this by a weight matrix W, adds a bias vector b1, then applies a ReLU (rectified linear unit) function to get the activation of the node. The final activation of the root node is then multiplied by another weight matrix U and added to another bias vector b2. Applying softmax gives probability scores for each label at the root node. Cross-entropy loss was then computed, with additional penalty given to large weights in the form of adding the l2 loss from W and U. Training was done using the generic TensorFlow GradientDescentOptimizer with a learning rate of 0.01, over a maximum of 40 epochs. The learning rate was annealed by 1.5 whenever the loss for an epoch was greater than 0.99 times the previous epoch loss.

4 Data Set and Evaluation

The dataset used was the original Convote corpus made available by Thomas et al. The original corpus consists of speech segments from Congressional floor debates of 2005. Each segment was labeled to specify the bill under discussion, the speaker, the speaker's party, whether the speaker directly mentions the bill in their speech, and how the speaker eventually voted on the bill. I first used the generic Stanford Constituency Parser (from Natural Language Toolkit's Stanford library) to construct syntax trees for each sentence in a random subset of the thousands of speeches in the corpus. I then filtered these trees in various ways as I tested the model, explained further below. The final set of trees was then split into training (70%), test (20%), and development (10%) with an approximately equal number of "voted yes" and "voted no" trees in each set. The performance of the network on each dataset was evaluated in terms of the accuracy of predictions on the unseen trees in the test set.

I first filtered the tree corpus to only include those that corresponded to sentences used in Iyyer et al.'s "ideologically biased" subset of the Convote dataset. From this set of about 6,000 trees, split almost evenly into "voted yes" and "voted no" trees, training was run over 40 epochs with 700 trees in the training set, 200 in the test set, and 100 in the development set. Using randomly initialized 35-D embeddings, training accuracy reached 57%, while test accuracy was only 47% (results summarized below). Using pre-trained 50-D GloVe embeddings, training accuracy reached 61%, with test accuracy of 53.6% (improvement stopped after 10 epochs). Using pre-trained 100-D GloVe embeddings, training accuracy reached 58.4%, while test accuracy reached 54.0%.

The ideologically biased sentences, while obviously relevant for determining the ideology of the speaker, were perhaps not the most helpful in determining the speaker's support for or opposition to a piece of legislation. The use of ideologically biased terms could certainly be used in either support of or opposition to a bill, depending on the content of the bill, which party proposed it, and the relevance of the issue to the particular speaker. To form a more salient dataset, I took a set of 18,000 parsed utterances from the original Convote dataset and took only ones that included the word "support" or "oppose" and did not include the word "yield" (thereby excluding irrelevant formalities such as "I yield the balance of my time"). This "extreme" dataset consisted of about 1,000 trees, split almost evenly between "voted yes" and "voted no" trees. After 40 epochs, the network reached 91% accuracy on the training set and 55.4% accuracy on the test set.

I also considered the effects of the content of the bill up for debate on the network's ability to model support/opposition in speech segments. To eliminate difficulties that might arise in training on a few bills, but testing on others, I created a small dataset of about 600 trees that all concerned the same bill. The network reached 77% accuracy on the training set and 65% accuracy on the test set after 3 epochs (early stopping threshold was reached) and 99.3% accuracy on the training set and 61.0% accuracy on the test set after 40 epochs.

5 Results

Dataset	Trees	Embedding	Training Accuracy	Test Accuracy
"Ideologically biased" Convote	3395 "voted yes" 2936 "voted no" (split 700 train, 100 dev, 200 test)	Randomly initialized and updated during training, dimension 35	57.0%	47.0%
"Ideologically biased" Convote	3395 "voted yes" 2936 "voted no" (split 700 train, 100 dev, 200 test)	Pre-trained GloVe, dimension 50	61.0%	53.6%
"Ideologically biased" Convote	3395 "voted yes" 2936 "voted no" (split 700 train, 100 dev, 200 test)	Pre-trained GloVe, dimension 100	58.4%	54.0%
"Extreme" Convote (must include "support" or "oppose")	499 "voted yes" 489 "voted no" (split 700 train, 88 dev, 200 test)	Pre-trained GloVe, dimension 100	91.0%	55.4%
Single debate from Convote	217 "voted yes" 370 "voted no" (split 300 train, 50 dev, 100 test)	Pre-trained GloVe, dimension 100	77.0% (3 epochs) **99.3% (40 epochs)**	**65.0% (3 epochs)** 61.0% (40 epochs)

Each new manipulation of the dataset gave slightly better results. Higher-dimension pre-trained word embeddings seem to have allowed the network to learn some aspect of the semantic relations between words, giving rise to higher test accuracies. Moving from the "ideologically biased" dataset of Iyyer et al. to a set of more salient "support" / "oppose" trees also improved the accuracy of the system. After 40 epochs, the network learned the training set very well (91%) but, most likely due to overfitting, did not perform well on the test set (55.4%). This may be because of an unequal distribution of speakers, parties, and particular bills in the training and test sets, such that the network was trained on a few speakers, parties, or bill debates, but was tested on others.

The network performed best when given a single debate as input. After just three epochs, test accuracy reached 65%, the highest test accuracy ever reached during the course of the project. After these promising results, I ran the network for a full 40 epochs, which seems to have pushed the network into overfitting to the training data: training accuracy was almost perfect (99.3%), while test accuracy dropped to 61.0%. The network most likely learned some pattern in the training data that was not as prevalent in the test data. This could be due to the fact that in such a small dataset, there was a relatively high probability that the training set would contain one set of speakers who perhaps spoke in similar ways or were directly responding to each other, while the test set contained a completely different mix of speakers who did not use similar sentences. However, these results are an indication that it is perhaps not the neural architecture that contributed to the poor test accuracies on previous iterations, but rather the nature of the dataset.

Overall, the intention of this project was for a recursive neural architecture to leverage syntactic structure and semantic relationships between words to perform sentiment analysis in the form of support/opposition classification. Ultimately, the network was able to do this reasonably well (65% accuracy) but did not match performance in similar previous work (Thomas et al. reached 70% for document-level classification). The network seems ultimately to have learned some aspects of syntactic structure, but certainly not to an extent that allowed generalization from training to test sets without a significant drop in performance. This task seems to be more difficult than the detection of political ideology done by Iyyer et al., which makes sense given that the same ideologically biased phrases could be used in either support or opposition of a piece of legislation, depending on the content of the bill, the party of the speaker, and the general political context.

6 Alternative Methods, Future Work

The model did not outperform Thomas et al.'s original performance using SVMs, nor did it achieve comparable test accuracy to the recursive neural networks used by Socher et al. or Iyyer et al. This is most likely due to the nature of the inputs to the model, which I tried to vary in the different datasets described above.

A recursive neural network can be more effective at sentiment analysis when labels are provided at the phrase level as well as the sentence level. This would allow the network to learn more detailed information regarding the structure of the speech segments. For example, negation before a phrase marked as positive should make the next node up in the tree be marked as negative. These distinctions are more difficult to make when the syntactic trees are labeled only at the sentence level. This may contribute to the network's difficulty in carrying out this task.

Beyond the variations on the speech transcripts described above, I would have liked to also explore representing more information about the context of the speech segment and the bill being discussed. The Convote dataset gave each bill under discussion a unique 3-digit code. Matching these codes with the actual bills, presumably available as public records online, I would have been able to include a representation of the bill in question as part of the input. This could have taken the form of a bag-of-words unigram feature vector representing the contents of the bill, a one-hot vector denoting the party sponsoring the bill, or a many-hot vector indicating which Congresspeople initially brought the bill to debate. Other information about the speaker could have been included as input, including their party, whether or not they mentioned the bill directly in their entire speech, what state they were from, etc. These would certainly be interesting directions to pursue, and could have helped to reveal more about what the network is able to learn about the speech segments when making the support/opposition decision.

To further explore the linguistic nature of the information learned by the network, another area of interest would be to test the network's performance on linguistically relevant structures. For example, it would be interesting to know whether the network could distinguish negation before a phrase as flipping the sentiment value of that phrase. Of course, this may be difficult given the lack of phrase-level sentiment labels, but this is certainly an interesting direction to pursue.

Finally, trying out different neural architectures on this task may help determine what aspects of linguistic structure are able to be learned by a neural network in the course of this task. Perhaps the constituency tree was not the most useful form of input, and a sequential recurrent neural network model might perform better. The recursive neural network ended up with fairly decent performance, but there clearly remains a lot of work to be done in further exploring this task.

References

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proc. of ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP* 2013, pages 1631–1642.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *EMNLP*.