# Prediction Political Parties in Context of Twitter

**Yoon Sung Hong**
UC Berkeley
Berkeley, CA 94709
yoonsung.h@berkeley.edu

**Jemima Shi**
UC Berkeley
Berkeley, CA 94709
jshi10@berkeley.edu

**Eric Sung**
UC Berkeley
Berkeley, CA 94709
eric.ju.sung@berkeley.edu

**Cheolho Jang**
UC Berkeley
Berkeley, CA 94709
cjang000@berkeley.edu

## Abstract

Modern techniques introduced by Facebook's InferSent research allows more efficient and effective approaches to natural language processing problems. In this project, we utilized various model architectures including Bi-RNN with a multilayer perceptron decoder and Bidirectional Encoder Representation from Transformers in order to analyze politically affiliated Twitter and news media data. The best results were ultimately obtained by pairing a multilayer perceptron decoder with the InferSent model as the encoder. We believe that we got this result due to 1) the implementation of additional hidden layers which effectively addressed the high dimensionality of each embedding, and 2) hyperparameter tuning such as adding dropout layers and weight decay in order to prevent overfitting. Through our analysis, we also find the evidence for the real world issue of partisan polarization of political discourse used in the context of social networks and online news media.

## 1 Introduction

Popularized in 2013 by Mikolov et al., word embedding uses predefined vector space to provide a new and unique way of dealing with many Natural Language Processing (NLP) problems in regards to semantic relationship between words. Starting with word embedding, the use of machine learning in solving semantic language problems has evolved considerably. The primary inspiration of this project is a 2015 paper by Severyn and Moschitti that focused semantic analysis in Twitter via convolutional neural networks. We take this conceptual approach one step further by introducing the use of bidirectional recurrent neural networks, a form of generative deep learning that simultaneously takes into account the backwards and forwards states of hidden layers. The model also takes into consideration concepts from Facebook's supervised sentence embedding learning tool InferSent. For the purpose of this project, we utilized our own version of approximately what InferSent accomplishes, but adjust the model appropriately in order to fit the context of Twitter.

In this project we also take our technical framework and apply it to answer real world questions on political news reporting. According to Pew Research Center, the American public's view of political issues are increasingly more partisan. This political polarization is at least partially fueled by the concept of "fake news", or the idea that the media is biased against particular political ideologies. We will examine this question through the lens of political discourse on the internet, see if the perceived political polarization in news media holds any statistical ground, and then attempt to use our data to predict the political preference of the author of a body of words based solely on their use of language. We do so by training our model on tweets from American politicians, who have explicit party association, validating our model accuracy to confirm the disparity in the language used by the two opposing political parties, and testing our model on politically polar news sources such as

Fox News and The Washington Post to see if there is a covariate shift between the language used by politicians on Twitter and the language used by news sources.

## 2 Technical Background

### 2.1 Word Embedding

#### 2.1.1 Word2vec and Skip-gram model

Word embedding is a set of techniques that can quantify words and phrases into numeric vectors. Word2vec is a word embedding toolkit invented by a Google team Mikolov, et al. (2013) and Word2vec can implement either the continuous bag-of-word (CBOW) architecture or the continuous skip-gram architecture. The continuous skip-gram architecture, which this project focuses more on, uses the word of interest (target word) to predict other surrounding words (context words). Skip-gram is slower than CBOW, but is advantageous in its accuracy with infrequently used words. Given T number of words, word vector $w_t$, limits on $-c$ and $c$, the goal of the skip-gram model is to find and maximize the average of the log of probability distribution surrounding the target word across the limits:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c} logp(w_{t+j}|w_t)$$

The hidden layer can be found through the function:

$$h = W^T x := \upsilon_{W_I}^T$$

And the output layer can be calculated using:

$$u_{c,j} = u_j = (\upsilon'_{W_j})^T h$$

The output layer involves computation of $c$ multinomial distribution such that the output layer weights matrix $W'$. Softmax with the following notation is used for activation:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})}$$

#### 2.1.2 Doc2vec

Introduced in 2014 by Mikilov and Le, the doc2vec method represents a natural evolution over word2vec (Mikilov et al., 2014). Doc2vec can take documents of varying length and creates a numeric representation of the document. It accomplishes this task by adding a paragraph vector to train document vector D, which takes into account the numeric representation of the document, along with the already established training of the word vectors W. This was one of the first implementations of a document/sentence embedding that used word embeddings as a basis model. Like its predecessor Word2vec, Doc2vec can be implemented in two different architecture designs: Distributed Memory version of Paragraph Vector (PV-DM) or Bag of Words version of Paragraph Vector (PV-DBOW).

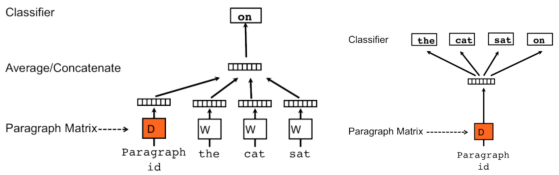


Figure 1: Graphic representation of PV-DM model (left) and PV-DBOW model (right) (Le et Mikolov., 2014)

### 2.2 Bidirectional RNN and Bidirectional LSTM

**Bidirectional RNN** Bidirectional recurrent neural networks (Bi-RNN) are a form of generative deep learning that simultaneously takes into account the backwards and forwards states of hidden layers by connecting the two directionalities into a single output. Bi-RNN allows both past and future information to be used and are thus more flexible in terms of input data, making them particularly useful for applications such as natural language processing where the context (i.e. word/phrase before and after the word of interest) of the input is essential. Bi-RNN can be interpreted as the simultaneous combination of two independent RNNS. This also allows us to create an overall phrase embedding representation after inputting multiple words. We do so by taking the last hidden layer of the encoder structure in both directions and concatenating the two of them. This will be the feature we will take advantage of in order to create sentence/tweet embeddings.

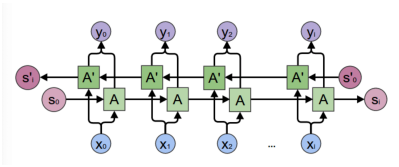

Figure 2: Structural Representation of Bidirectional Recurrent Neural Networks (Olejnik, 2017).

**Bidirectional LSTM** Bidirectional LSTM takes a similar approach as Bi-RNN and its general architecture follows the diagram on Figure 2. Given a sequence of words $w_t(t = 1, ..., T)$ in a sentence, where T = number of words, a Bi-LSTM computes T vectors of $h_t$. For each $h_t$ for $t = 1, ..., T$, the vector represents the concatenation of the forward and backward LSTM that read the sentences in both forward and backward.

$$\overrightarrow{h_t} = \overrightarrow{LSTM_t}(w_1, ..., w_T)$$
$$\overleftarrow{h_t} = \overleftarrow{LSTM_t}(w_1, ..., w_T)$$
$$h_t = \overrightarrow{h_t}\overleftarrow{h_t}$$

Bi-LSTM can be encoded through max pooling or mean pooling that generates a set of $n$ vectors for $n$ words in a sentence, where each vector is a concatenation of the forward and backward LSTM. Max pooling takes the maximum value of the dimensions of the hidden vectors, while mean pooling takes the average. We will look at a working version of max pooling Bi-LSTM when discussing InferSent later.

### 2.3 Bidirectional Encoder Representation From Transformers (BERT)

Opened up to the public in late 2018, BERT is Google's deep learning algorithm for natural language processing. BERT's ability to fine-tune with limited amounts of data on a pre-trained model allows it to have significantly higher performance compared to other comparable models. Before BERT, a prominent constraint with current language models was its unidirectionality, such as OpenAI GPT, which uses left to right transformer, and ELMo, which uses left to right and right to left LSTM.

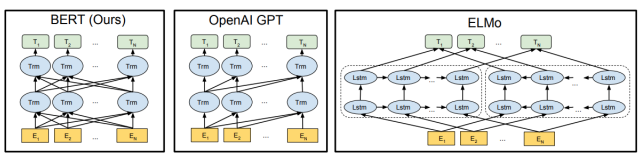


Figure 3: Design of Current Language Models(Middle, Right) and BERT(Left) (Devlin, et al., 2018)

**Masked Language Model** These restrictions can hinder the architectural choices of pretrained datasets. BERT answers this problem with the "Masked Language Model," where random tokens are purposely hidden for the model to figure out with only the words in context, allowing words from both left and right to enter contextualization. The net result of this process creates bidirectionality.

### 2.4 Multilayer Perceptron

**Background** A single perceptron takes a set of inputs and uses a set of weights and a bias term to compute the output in the following formula:

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(w^T x + b)$$

A multilayer perceptron has multiple cases of this computation that build on top of one another to create layers. The perceptron begins with the input computation which goes through hidden layer(s) and creates outputs. Each layer is activated with functions such as the ReLU or the Sigmoid function to enforce nonlinearity between layers.

### 2.5 InferSent

Facebook's Infersent research aimed to address word embedding's shortcomings on dealing with large chunks of text. Results showed that "an encoder based on a Bi-LSTM architecture with max pooling, trained on the Stanford Natural Language Inference (SNLI) dataset" provided optimal accuracy. The NLI tracked sentence entanglement by classifying text relationship into three separate classes: Entailment, Neutral, Contradiction. There were different encoder architectures implemented, including LSTM, GRU, Hierarchical Convnet, and Bi-LSTM with max pooling and mean pooling. This was then decoded and classified into three classes using a multi-layer perceptron with one hidden layer of 512 units. We aim to adapt InferSent's approach by implementing their encoder architecture (particularly the Bi-LSTM with max pooling) and building our own multi-layer perceptron model for the decoder/classification model (in order to capture the full complexity of the tweets and their embeddings and to reduce overfitting on training data).

**Basic NLI Training Scheme** The optimal model trains on SNLI through sentence encoding and Bi-LSTM-based models, which clearly distinguish the encoding of multiple sentences individually. This architecture uses an encoder which produces sentence vector representation for the premise $u$ and hypothesis $v$ for each sentence. Afterwards, three different matching methods between u and v are applied: concatenation $(u, v)$, element-wise multiplication $u \times v$, and element-wise absolute difference $|u - v|$. The final product vector, which accounts for both $u$ and $v$, is then fed into multiple fully connected layers and finally the softmax activation layer.

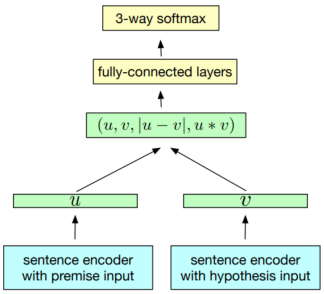


Figure 4: Generic NLI training scheme as represented in Conneau, et al. (2017)

**Bi-LSTM Max Pooling** The approach uses an encoder based on Bi-LSTM architecture with max pooling, where the final representation uses the maximum value of the dimensions of the hidden vectors (Alexis et. al, 2017). Each concatenation $h_t$ can be formed to a fixed size vector by "selecting the maximum value over each dimension of the hidden units" in a process named max pooling.

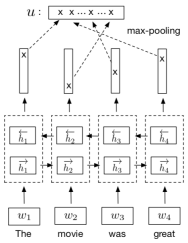


Figure 5: Bidirectional LSTM max pooling network as represented in Conneau, et al. (2017)..

# 3 Methods

## 3.1 Data Collection and Preprocessing

The training/validation data set included tweets from Democratic and Republican senators, House representatives, and governors spanning the beginning of 2017 to early 2019. The testing dataset included headlines and first paragraphs from political articles posted on presumably liberal/conservative leaning news websites posted from mid February to April of 2019. All of the data were collected either through manual scrapings or calls to the Twitter API, and then reviewed to ensure that they were political in their topics of discussion. The datasets were then cleaned to account for duplicate entries, unidentifiable characters, and the recency of each entry.

## 3.2 Bi-RNN with Dense(2) Layer

First, the sentences were tokenized into words. We found the maximum number of words used in a tweet, and assigned it to the variable $m$. We then padded any tweets with additional <unk> tokens up until the total number of tokens in the sentence was equal to $m$, if the number of words in a sentence was less than $m$. Following this, similar to previous researches such as Facebook's InferSent, pre-trained word embeddings were used on the tokenized tweets to generate word embeddings for each tweet. We specifically used GloVe's word vectors trained on Twitter. Word vectors trained on the Twitter corpus correlate more closely to the language used in Twitter and we assumed that this would allow us to increase our model accuracy by preserving more words in our training and testing data.

Following the generation of word embeddings, we created a Bidirectional Recurrent Neural Network class to be used to generate our sentence embeddings from these inputted word embeddings. An LSTM model's encoder was set up with embedding layers of size 200 and 4 hidden layers of size 100 to capture the full complexity of words used by politicians in tweets. The generated embeddings were then decoded using a dense layer with one hidden layer of 2 units. Gluon's absence of hybridization feature on the RNN layers resulted in the training taking over 15 minutes per epoch, limiting our decoder implementation to use a Dense(2) layer instead of a more complex architecture. Before training, we used Xavier initializer for initializing the weights, as this ensured that the neuron activation functions are not starting in dead regions. We also used Stochastic Gradient Descent instead of Adam as our optimizer as it was achieving higher training accuracy even after multiple epochs. We used learning rate of 0.2 with the optimizer function. We trained our model for 150 epochs. With these parameter choices, $x$ will be updated in a following way:

$$x \leftarrow x - \eta \nabla f_i(x).$$

## 3.3 Bi-RNN InferSent

Similar to the approach addressed above, we used pre-trained word embeddings on each tweet. We used GloVe's word vectors trained on Common Crawl data, but with a different assumption that was the vocabulary used by the politicians is relatively similar to the traditional English language and therefore unparallel to the text used in Twitter.

We used the pre-trained word embeddings as well as InferSent's Bidirectional RNN (LSTM) encoder to generate 4096-dimensional embeddings per tweet. One of the advantages of InferSent was the

way in which it built the encoder and decoder separately; leveraging this, we used the embeddings outputted from the encoders but improved upon the Multilayer Perceptron model used by the original research paper, a Multilayer Perceptron with a single 512 unit hidden layer to decode and predict labels. In our approach, we believed that the model should be more sophisticated to be capable of fully capturing the complexity from the high dimensionality per embedding. Therefore, we implemented a total of three hidden layers: one with 2048 units, another with 512 units, and finally one with 2 units for binary classification. The first two layers had ReLU as their activation functions, with an additional dropout layer added in between set to a probability of 0.25. We also added a weight decay parameter of 0.001, learning rate of 0.2, and learning rate decay of 0.85 every 4 epochs to combat overfitting as it trained a total of 150 epochs until total loss dropped to a desirable value.

We experimented with each parameter to find the optimal combination. We tried weight decay values between 0.0005 and 0.0015 in increments of 0.0001 and determined that 0.001 reduced overfitting most effectively while still simultaneously maintaining high final training accuracy. Similarly, we experimented with our learning rates, trying out values ranging from 0.05 to 0.5 in 0.05 increments. We learned that initializing the model with a larger learning rate, in our case 0.2, in conjunction with weight decay achieved the highest training accuracy. However, as a trade-off, this also led to higher fluctuations in loss and accuracy in between epochs. To counteract this side-effect, we tried adjusting the weight decay value between a range of 0.2 and 0.9 in increments of 0.05 alongside various fixed decay periods before ultimately deciding on values of 0.85 and 4 respectively.

### 3.4 BERT

Unlike our attempts with Bidirectional RNN models, the BERT model uses a model pre-trained on Wikipedia and the BookCorpus. We assumed that the more formal structure and nature of political tweets would be able to match the corpus and fine-tune a more accurate model. The model padded each input to a sequence of 128 uniform tokens which were then fed in batches of size 32 alongside a learning rate of 2e-5. We appended a final dense layer with weights equal to the number of pooled layers and a dropout of 0.9 that would ultimately perform the final classification of 0 (Republican) or 1 (Democratic) for the tweets and news headlines respectively.

### 3.5 Testing Validation

We tested our data using two different methods. First, we split our training data into training and validation datasets using a ratio of 8:2. We used this training and validation dataset to fine-tune our models, adjusting their parameters and optimizer functions to achieve the best results. Second, after our model adjustments were complete, we used the test dataset from the news sources' headlines to assess our testing accuracy on the model trained on Twitter data.

## 4 Results

Below are the results with our training, validation, and test dataset with the 3 methods implemented.

The InferSent's encoder method in conjunction with a customized multilayer perceptron achieved the best results, outperforming the Facebook research's encoder-decoder architecture. We suspect that this is attributed to the custom multilayer perceptron's complex structure and preventative measures for overfitting, such as a dropout layer and weight decay, which allowed the model to surpass the original multilayer perceptron in both training and validation accuracy.

Another important result is the out-performance by the InferSent Bi-RNN model with custom MLP (Model 3.3) over the Bi-RNN with a dense layer of size 2 (Model 3.2) as the decoder. Although model 3.2 did not have as sophisticated of a decoder structure as model 3.3 did, it used GloVe's word embeddings trained on Twitter corpus, which we would anticipate to give more accurate semantic representations of these tweets. The difference in accuracy is likely for the two following reasons:

1. The Twitter embeddings by GloVe were created in 2014. The corpus is likely outdated and thus not able to fully capture the semantics of today's politicians accurately.
2. The language used by politicians, even on the platform of Twitter, is usually formal and unerring in grammar and spelling. Consequently, using non-Twitter-central word embeddings which offer more traditional semantic representations proves to be more effective.

| Model | Accuracy | | | Word Embeddings Used |
|---|---|---|---|---|
| | Training | Validation | Test | |
| Bi-RNN with Dense(2) as Decoder | 0.92 | 0.77 | 0.54 | glove.twitter.27B.200d |
| InferSent Bi-RNN with MLP as Decoder (one 512 units hidden layer) (no dropout, weight decay, or learning rate decay) | 0.86 | 0.78 | 0.51 | glove.840B.300d |
| InferSent Bi-RNN with custom MLP as Decoder, with dropout, weight decay, and learning rate decay | **0.94** | **0.86** | 0.54 | glove.840B.300d |
| BERT | 0.8 | 0.73 | **0.6** | wordpiece embeddings |

Figure 6: Results from different models.

Overall, all models performed poorly in classifying the political parties of the news articles, with BERT having the highest testing accuracy of the four at 0.6. However, BERT performed most poorly in classifying the training and validation datasets and we believe that this is likely due to the embeddings used by BERT, which have neither the contextual understanding of politics nor Twitter.

## 5 Covariate Shift Correction

All of our models fail to accurately predict the political preferences of polarized news sources, confirming that there is a clear covariate shift between the training and news media test datasets; this was to be expected, since the language used in news media articles is extremely different to that of politicians on social media sites such as Twitter.

We attempted implementing a covariate shift correction on our InferSent model with custom MLP, using the following identity:

$$\int p(x)f(x)\,dx = \int p(x)f(x)\frac{q(x)}{q(x)}\,dx = \int q(x)f(x)\frac{p(x)}{q(x)}\,dx$$

We obtained $\frac{p(x)}{q(x)}$ by using the train and test as labels in a mixed dataset. Following the identity above, we used logistic regression to re-weight our training data and used these pre-calculated weights for training on our labels. The results improved, but still managed to only achieve 60% test accuracy after 100 epochs.

We also attempted to mix a portion of our test data into the training data to see if the model would be able to distinguish on its own the features that were specific to the test data. We did this by taking 20% of our test data (around 300 articles) and adding it into our training. Unfortunately, the test accuracy did not improve significantly with this alteration either, increasing slightly to 54%. We suspect this was due to a considerable difference in scale; the addition of 300 rows of test data to our training set with over a million rows is not substantial enough for the model to recognize the test dataset's unique feature patterns.

Finally, we attempted to extract the most differentiating features by building a random forest classifier with the intial labeling as train and test again. Due to the computational limitations and the large dimensionality of the embeddings (4096) we applied this to a small portion of our training data, sampling 10,000 data points. With this method, we identified 96 features which worked as the best split points and excluded them as features altogether. Then used our InferSent Bi-RNN with custom MLP in the same manner as before on the filtered 4000 features. This method decreased the training accuracy from 94% to 90%, but increased the test accuracy to 65%. We suspect this is because the removal of these features made the language used between politicians and news media more homogeneous.

Considering how we are building our model as a binary classifier, a test accuracy of 65% is still a poor result. We were only able to test out a limited number of approaches to try to fix our covariate shift problem; more time, additional methods, and more intensive data cleaning measures such as vocabulary substitution would have likely helped further improve our test accuracy.

## 6 Conclusion

It is clear that there is a distinction in language used between Republican and Democrat politicians. The InferSent model with custom MLP was able to successfully classify the parties with over 94% accuracy even after taking measures to prevent overfitting.

These were some limitations to this project. To list a few:

1. Our Bidirectional RNN model with Dense(2) as a decoder was built solely on Gluon, which does not offer hybridization of RNN neural nets. This put our training speed at an average of 1080 seconds per epoch, not only limiting but also deterring us from attempting to build a more complex decoding architecture.
2. InferSent only offered the options of GloVe's Common Crawl and fastText as word embeddings. Although InferSent's encoder architecture allowed for significantly faster training times, it would have been helpful to use GloVe's Twitter word embeddings and compare the results under the same encoder/decoder architectures as one in model 3.3.
3. For our data, we included retweets as part of the politicians tweets data. We did this in order to ensure that we have enough data to fit and test our models. Additionally, the Twitter API restricted collection to roughly 8000 tweets per politician. This meant that we may have missed quite a large proportion of tweets from some of these politicians, even after narrowing it down to tweets dated from 2017 on-wards (For example, internet-active politicians such as Congresswoman Alexandria Ocasio-Cortez tweet and retweet more than 10 times a day). It would have helped all of our models' performances if we only used the original tweets and been able to collect all the tweets made by these politicians without the restriction quota set by Twitter. This would have both reduced the noise and given us a sufficient amount of data for optimal model performance.

Due to the limitations in time and resources, we ended up developing our models with these issues unresolved. Looking forward, it would help model performances to address these issues more effectively.

Overall, it is clear that politicians from opposing parties speak the language of Twitter very contrastingly, even when it comes down to similar topics. The politicians within the same parties have a homogeneity in their language to an extent where a neural-network model is able to classify the difference between the two parties as a whole. Pre-trained embeddings prove to be effective in preparing the tweets to be semantically examined in the neural network models, and the general word embeddings such as GloVe Common Crawl word embeddings performed better than those trained specifically on Twitter corpus due to the officiality of the vernacular used by the politicians. Even though these word embeddings do not allow us to semantically process politically-specific words such as 'Obamacare' and 'Mueller Report', the semantics surrounding these words are able to convey enough information for an accurate classification of their political associations.

## References


[1] Bickel, Steffen, et al. *Discriminative Learning Under Covariate Shift*, Journal of Machine Learning Research, 10, 2137–2155, 2009.

[2] Conneau, Alexis, et al. *Representations from Natural Language Inference Data*, Conference on Empirical Methods on Natural Language Processing (EMNLP), 2017. https://research.fb.com/publications/supervised-learning-of-universal-sentence-representations-from-natural-language-inference-data/

[3] Delisle, Laure, et al. *Troll Patrol Methodology Note*, Amnesty International, 2018. https://decoders.azureedge.net/data-viz/images/Troll

[4] Devlin, Jacob, et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*, *arXiv preprint arXiv:1810.04805v1*, 2018.

[5] Le, Quoc, and Tomas Mikolov. *Distributed Representations of Sentences and Documents*, *arXiv preprint arXiv:1405.4053*, 2014.

[6] Mikolov, Tomas, et al. *Distributed Representations of Words and Phrases and their Compositionality.*, In NIPS , 3111–3119. : Curran Associates, Inc., 2013.

[7] Severyn, Aliaksei, and Alessandro Moschitti. *UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification.*, Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), 2015.

[8] Tyson, Alec. In *Views of Mueller's investigation – and Trump's handling of the probe – turn more partisan* PEW Research, 2018. https://www.pewresearch.org/fact-tank/2018/09/24/views-of-muellers-investigation-and-trumps-handling-of-the-probe-turn-more-partisan/

[9] Xiao, Lizhong, et al. *Research on Patent Text Classification Based on Word2Vec and LSTM.*, 2018 11th International Symposium on Computational Intelligence and Design (ISCID), 2018. https://ieeexplore.ieee.org/abstract/document/8695493

[10] Zhou, Peng, et al. "Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling", COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3485–3495, 2016. https://aclweb.org/anthology/C16-1329