Opinion Tagging Using Deep Recurrent Nets with GRUs

Alex Adamson
Stanford University
aadamson@stanford.edu

Vehbi Deger Turan
Stanford University
vdturan@stanford.edu

Abstract

Opinion tagging is an important task when distinguishing features of a product or a service in reviews, or filtering and summarizing a large body of text for personal perspectives. We strive to attain the performance level of conditional random fields for structural prediction of agent, attitude and target tags using neural networks. The models we constructed and evaluated include a softmax regression baseline, a deep bidirectional recurrent network, and an extension using gated recurrent units instead of ReLU nonlinearities for the bidirectional RNN. For more complicated classifications where local dependencies are not present, the GRU model outperforms DRNN by a small margin. It is important to acknowledge that development classification tags are not objective and debatable by human classifiers, since the subject is quite complex and above concrete syntactic features. However, for simpler tasks such as opinion agent labels, DRNN performs significantly better.

1 Introduction

For events in which many members of a community release statements concerning a single subject, the volume of responses and their unstructured nature can be an obstacle to product developers, policy makers or other interested parties getting a true sense of the attitudes of a public body. We intend to develop an opinion tagging model using neural networks that, given a corpus of documents, can tag sequences in the text as being expressions of an opinion (an attitude), being the holder of an opinion (an agent), or being the target of an opinion. To limit the scope of the problem and to make it more tractable for neural network-based classifiers, we do not plan to capture the relationships between holders, opinions and targets (i.e. we do not plan to tag a certain sequence that is an opinion as being held by a particular holder within some tagged holder sequence).

The fine-grained opinion mining task seeks to, given textual input, find the opinions expressed therein, their intensity and other properties such as their holder and their target. In particular, we first explore the fitness of a deep bidirectional recurrent network for this task using the architecture described in Irsoy and Cardie 2014 and then attempt to extend it by using gated recurrent units instead of ReLU units.

2 Related Work

2.1 Opinion mining

Much of the previous work in opinion mining focused on extracting subjective expressions and typically used conditional random fields for sequence tagging. Much of this work focused on developing token- and phrase-level feature extraction (such as part-of-speech tagging and token-level sentiment tagging). Choi et al. 2005 employed conditional random fields to jointly identify opinions and opinion holders. Yang and Cardie 2013 develops a conditional random field to jointly infer opinion expressions, opinion holders and opinion targets by adding learned relation features. Some of these models have proved highly successful on the opinion analysis task. In particular, the joint inference model from Yang and Cardie 2013 achieves overlap F1 scores above 60% on the MPQA 2.0 dataset for all classes of interest (opinion expressions, opinion holders and opinion targets).

There is prior work on recurrent neural networks on opinion mining as in Irsoy and Cardie 2014 that is limited to discriminating null-class text from subjective expressions. We would like to extend it to discriminate between additional opinion-related classes, such as opinion target, holder and attitude.

2.2 Recurrent neural networks

Recurrent networks have been a mainstay of deep learning for some time. The typical Elman-type network (Elman 1990) has no depth-in-space and passes the result of the hidden layer at timestep t h_t to the next temporal hidden layer, h_{t+1}. Each layer takes as additional input a representation of some token (x_t) in a sequence, and may output a tag for the token y_t. This construction is limited for language-based tasks since typically a token's tag can be informed by context from the future instead of only the past.

2.2.1 Bidirectional recurrent neural networks

This observation motivates the introduction of a bidirectional variant (Schuster and Paliwal 1997) that has at each timestep a future-informed hidden unit $\overleftarrow{h}$ and a past-informed hidden unit $\overrightarrow{h}$. The output y_t is then typically a function of both $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$.

2.2.2 Stacked recurrent neural networks

Another limitation of the Elman-type network is its lack of depth-in-space: the single spatial layer restricts its ability to learn complex abstractions over the sequence. This motivates the proposal of networks using stacked hidden units (El Hihi and Bengio 1995) where the hidden unit in layer i at timestep t $h_t^{(i)}$ propagates its output to its temporally next neighbor ($h_{t+1}^{(i)}$) and its spatially next neighbor ($h_t^{(i+1)}$).

2.2.3 Stacked bidirectional recurrent neural networks

The models we expand upon in this work are a fusion of the two extensions to Elman-type networks introduced above. Irsoy and Cardie 2014 introduces a recurrent network with depth-in-space and bidirectionality and applies it to subjective expression tagging.

2.3 Gated Recurrent Units

Previous proposals has been made to extend recurrent neural networks by replacing the simple non-linearities performed by the hidden units with more complex gated functions. The intuition behind the work has been to introduce gating in hidden units to allow them to operate at different time-scales by selectively remembering and forgetting their state based on input signals (Hochreiter and Schmidhuber 1997, Chung et al. 2015).

3 Methods

Here we describe the network architectures we propose for use in the opinion tagging task.

3.1 Recurrent neural network with depth in space and bidirectionality

Traditional Elman-type networks use depth in time in that they employ a single feedforward hidden unit for each token in the input sequence.

$$\begin{aligned} h_t &= f(Wx_t + Vh_{t-1} + b) \\ y_t &= g(Uh_t + c) \end{aligned}$$

At each timestep, the same transformation W is applied to the input and the same transformation V applied to the previous timestep's output. Hence, the network all hidden representations h_t must lie in the same representation space, limiting the ability of the network to form abstractions over previous hidden representations as they move forward in time.

Additionally, a particular hidden unit can only make use of abstract representations made over representations of tokens from the past. In tasks in which the input sequence is textual, this is clearly limiting because the output layer y has no information about the future context.

With these limitations in mind, we propose the use of a deep bidirectional recurrent neural network for this task. Each layer of the network can operate in a different representation space, forming abstract, hierarchical representations over its input (Bengio 2009). The model architecture that used here is taken from Irsoy and Cardie 2014: each layer computes computes its output using both forward and backward connections using the hidden representations computed at the previous layer as the input.

For $i > 1$, we have

$$\overrightarrow{h}_t^{(i)} = f(\underrightarrow{\overrightarrow{W}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overrightarrow{W}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)}) \tag{1}$$

$$\overleftarrow{h}_t^{(i)} = f(\underrightarrow{\overleftarrow{W}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overleftarrow{W}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t=1}^{(i)} + \overleftarrow{b}^{(i)}) \tag{2}$$

and for $i = 1$ we have

$$\overrightarrow{h}_t^{(1)} = f(\overrightarrow{W}^{(1)} x_t + \overrightarrow{V}^{(1)} \overrightarrow{h}_{t-1}^{(1)} + \overrightarrow{b}^{(1)}) \tag{3}$$

$$\overleftarrow{h}_t^{(1)} = f(\overleftarrow{W}^{(1)} x_t + \overleftarrow{V}^{(1)} \overleftarrow{h}_{t+1}^{(1)} + \overleftarrow{b}^{(1)}) \tag{4}$$

We only connect the last layer (which we denote layer L) to the output layer:

$$y_t = g(\underrightarrow{U} \overrightarrow{h}_t^{(L)} + \underleftarrow{U} \overleftarrow{h}_t^{(L)} + c) \tag{5}$$

3.2 Recurrent neural network with depth in space and bidirectionality using gated recurrent hidden units

One difficulty in the performance of recurrent neural networks is the inability to get recurrent neural networks to encode long-term temporal interactions between tokens in the input sequence. A naive approach has previously been employed wherein the nonlinearity between hidden units in the same spatial layer are modified and paths are added between non-neighboring hidden units to encourage gradients to propagate farther back than they otherwise would. Other methods such as long short-term memory (Hochreiter and Schmidhuber 1997) have been employed to give hidden units memory cells to store potential long-term dependencies that are then gated and used when a dependency is deemed to exist. Recently, gated recurrent units have been proposed (Cho et al. 2014) that adaptively remembers and forgets its state based on input signals. Chung et al. 2015 explores the usage of gated recurrent units to allow different spatial layers (and temporal connections between spatial layers) to operate at different time scales.

We propose a novel architecture that uses bidirectional spatial layers with gated recurrent hidden units. Unlike the gated feedback recurrent neural network, our architecture lacks global reset gates between consecutive time steps at different layers (or any forward propagation from higher layers to lower layers).

Formally, the model is defined as follows:

For $i > 1$, we have

$$\overrightarrow{z}_t^{(i)} = \sigma(\underrightarrow{\overrightarrow{Wz}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overrightarrow{Wz}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overrightarrow{Vz}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{bz}^{(i)}) \tag{6}$$

$$\overrightarrow{r}_t^{(i)} = \sigma(\underrightarrow{\overrightarrow{Wr}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overrightarrow{Wr}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overrightarrow{Vr}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{br}^{(i)}) \tag{7}$$

$$\widetilde{\overrightarrow{h}}_t^{(i)} = f(\underrightarrow{\overrightarrow{W}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overrightarrow{W}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overrightarrow{r}_t^{(i)} \circ \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)}) \tag{8}$$

$$\overrightarrow{h}_t^{(i)} = \overrightarrow{z}_t^{(i)} \circ \overrightarrow{h}_{t-1}^{(i)} + (1 - \overrightarrow{z}_t^{(i)}) \circ \widetilde{\overrightarrow{h}}_t^{(i)} \tag{9}$$

$$\overleftarrow{r}_t^{(i)} = \sigma(\underrightarrow{\overleftarrow{Wr}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overleftarrow{Wr}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{Vr}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{bz}^{(i)}) \tag{10}$$

$$\overleftarrow{z}_t^{(i)} = \sigma(\underrightarrow{\overleftarrow{Wz}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overleftarrow{Wz}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{Vz}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{bz}^{(i)}) \tag{11}$$

$$\widetilde{\overleftarrow{h}}_t^{(i)} = f(\underrightarrow{\overleftarrow{W}}^{(i)} \overrightarrow{h}_t^{(i-1)} + \underleftarrow{\overleftarrow{W}}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{r}_t^{(i)} \circ \overleftarrow{V}^{(i)} \overleftarrow{h}_{t=1}^{(i)}) \tag{12}$$

$$\overleftarrow{h}_t^{(i)} = \overleftarrow{z}_t^{(i)} \circ \overleftarrow{h}_{t+1}^{(i)} + (1 - \overleftarrow{z}_t^{(i)}) \circ \widetilde{\overleftarrow{h}}_t^{(i)} \tag{13}$$

and for $i = 1$ we have

$$\overrightarrow{z}_t^{(1)} = \sigma(\overrightarrow{Wz}^{(1)} x_t + \overrightarrow{Vz}^{(1)} \overrightarrow{h}_{t-1}^{(1)} + \overrightarrow{bz}^{(1)}) \tag{14}$$

$$\overrightarrow{r}_t^{(1)} = \sigma(\overrightarrow{Wr}^{(1)} x_t + \overrightarrow{Vr}^{(1)} \overrightarrow{h}_{t-1}^{(1)} + \overrightarrow{br}^{(1)}) \tag{15}$$

$$\widetilde{\overrightarrow{h}}_t^{(i)} = f(\overrightarrow{W}^{(1)} x_t + \overrightarrow{r}_t^{(1)} \circ \overrightarrow{V}^{(1)} \overrightarrow{h}_{t-1}^{(1)}) \tag{16}$$

$$\overrightarrow{h}_t^{(1)} = \overrightarrow{z}_t^{(1)} \circ \overrightarrow{h}_{t-1}^{(1)} + (1 - \overrightarrow{z}_t^{(1)}) \circ \widetilde{\overrightarrow{h}}_t^{(1)} \tag{17}$$

$$\overleftarrow{z}_t^{(1)} = \sigma(\overleftarrow{Wz}^{(1)} x_t + \overleftarrow{Vz}^{(1)} \overleftarrow{h}_{t+1}^{(1)} + \overleftarrow{bz}^{(1)}) \tag{18}$$

$$\overleftarrow{r}_t^{(1)} = \sigma(\overleftarrow{Wr}^{(1)} x_t + \overleftarrow{Vr}^{(1)} \overleftarrow{h}_{t+1}^{(1)} + \overleftarrow{br}^{(1)}) \tag{19}$$

$$\widetilde{\overleftarrow{h}}_t^{(i)} = f(\overleftarrow{W}^{(1)} x_t + \overleftarrow{r}_t^{(1)} \circ \overleftarrow{V}^{(1)} \overleftarrow{h}_{t+1}^{(1)}) \tag{20}$$

$$\overleftarrow{h}_t^{(1)} = \overleftarrow{z}_t^{(1)} \circ \overleftarrow{h}_{t+1}^{(1)} + (1 - \overleftarrow{z}_t^{(1)}) \circ \widetilde{\overleftarrow{h}}_t^{(1)} \tag{21}$$

We only connect the last layer (which we denote layer L) to the output layer:

$$y_t = g(\underrightarrow{U} \overrightarrow{h}_t^{(L)} + \underleftarrow{U} \overleftarrow{h}_t^{(L)} + c) \tag{22}$$

For convenience, we define the $h_0^{(i)}$ and $h_{T+1}^{(i)}$ to be the zero vector for all layers i.

For all experiments, we let g be the softmax function and f be $ReLU$ and $\tanh$ for the first and second architectures respectively.

4 Experiments

4.1 Hypotheses

We expect that that the recurrent neural networks with gated recurrent units will outperform those without. It is unclear if either model will achieve performance comparable to those seen in the conditional random field models. In particular, such models have access to parse trees and opinion lexicons. Our hope is that the models we employ will learn abstractions over the input representations that will give it similar information.

4.2 Data

We use the MPQA 2.0 dataset (Wiebe, Wilson, and Cardie 2005). The dataset consists of 15737 sentences drawn from the world English-language press. Each sentence is tagged by a trained human for features such as attitudes, attitude targets, attitude holders (agents), objective speech events, and various subjective statements.

We separate the data into a training set consisting of 12590 sentences, a validation set consisting of 1574 sentences, and a test set consisting of 1573 sentences. During training, we evaluate the models against the validation set during model selection.

4.3 Evaluation

We train each model to distinguish tokens of one tag from tokens of all other tokens, so for instance, the ground truth tag for a token used during training for the model intended to distinguish opinion expressions will be either positive or negative, positive meaning the token is part of an opinion expression and negative meaning it is not.

To evaluate the models, we employ proportional and binary overlap. The binary overlap metric counts all contiguous tags in a predicted sequence as correct if any part of the predicted sequence of contiguous tags overlaps a sequence of contiguous tags with the same label in the ground truth (Yang and Cardie 2013). Proportional overlap is the obvious metric: We count a tag as correct if it matches the ground truth. We use these soft metrics because, as Wiebe, Wilson, and Cardie 2005 notes, human evaluators have difficulty distinguishing the boundaries of expressions. Hence, exact overlap would be too limiting as in many cases inexact overlap is essentially correct. Recall, precision and F1 using these metrics are reported. Model selection uses proportional overlap F1 scores to determine which trained models are "best" for a given task.

4.4 Word Vectors

To form input sequences to the first layer of the models, we map each token to the corresponding the 300-dimensional GloVe embedding trained using the Common Crawl set (Pennington, Socher, and Manning 2014).

4.5 Regularization

When performing model selection on either model, we employ grid search over several values (including 0) of λ for L2-regularization. We found that neither model is prone to overfitting and that using L2-regularization was mostly useful to prevent overflow during training.

We also experiment with dropout (Hinton et al. 2012), a regularization technique that has been used to great effect when training large neural networks to prevent learned hidden features from co-adapting. We grid search over dropout rates of $0.0, 0.1$ and 0.2 during model selection. Validation error tended to be minimized with a dropout rate of 0.1.

4.6 Training

For the objective function, we employ the standard cross-entropy function. We use mini-batch stochastic gradient descent with momentum and a batch size of 80. Networks are trained for 80 epochs and early termination is used. Nearly all models terminate early (i.e., the best validation results are seen before the eightieth epoch). Every hidden layer receives a supervised error signal from the output layer even though only the last hidden layer is connected to the output layer. Without adding these error signals, the networks tend to converge to output label probabilities that are approximately the prior class probabilities.

When computing the error vector propagated back from the output layer, we multiply the element of the vector corresponding to the t^{th} timestep by a constant $\gamma \leq 1$ if the ground truth label for the input token at t was the null class (i.e. not the tag of interest). This technique was shown to improve performance on the expressive-subjective-expression and direct-subjective-expression extraction task in Irsoy and Cardie 2014. Choosing $\gamma < 1$ encourages the model to be more cavalier about labelling tokens as the tag of interest; We observed that $\gamma \geq 0.8$ led to very low recall scores on both architectures. During model selection, we perform grid search on γ over values $0.3, 0.5$ and 0.7.

5 Results

We test both architectures using four spatial layers.

5.1 DRNN vs. GRU

	Model	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
Agent	SR	0.422	0.426	0.333	0.482	0.333	0.452
	DRNN	**0.653**	**0.675**	**0.707**	**0.746**	**0.679**	**0.709**
	GRU	0.632	0.661	0.675	0.722	0.653	0.690
Attitude	SR	**0.339**	0.341	0.192	0.544	0.245	0.419
	DRNN	0.276	0.344	**0.625**	**0.733**	0.383	0.468
	GRU	0.291	**0.360**	0.575	0.672	**0.386**	**0.469**
Target	SR	0.230	0.230	0.040	0.125	0.069	0.162
	DRNN	0.243	0.280	**0.398**	**0.514**	**0.302**	0.362
	GRU	**0.266**	**0.320**	0.343	0.447	0.300	**0.373**

Table 1: Results for models from grid search over λ, dropout rate and null-class weight. All models have 25-dimensional hidden units and use the GloVe 300-dimensional Common Crawl embeddings.

Both architectures were able to beat the baseline softmax regression on essentially all tasks. Results are generally as expected — the use of the GRU slightly improves performance on most tasks, and particularly on the more difficult tasks of target extraction (binary F1 of 0.373 vs. 0.362) and attitude extraction (binary F1 of 0.468 vs 0.469). The DRNN architecture beats the GRU architecture by a wide margin on the agent extraction task. Agent sequences tend to be short and are typically identifiable based on local dependencies or token-level priors, so it is expected that the DRNN should achieve strong performance on the task, but it is unclear why the GRU architecture performed relatively poorly.

5.2 Additional hidden units

	Model	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
Agent	DRNN	**0.669**	**0.690**	**0.722**	**0.755**	**0.695**	**0.721**
	GRU	0.625	0.656	0.664	0.716	0.644	0.685
Attitude	DRNN	**0.308**	**0.371**	0.536	0.694	0.391	0.482
	GRU	0.292	0.364	**0.602**	**0.718**	**0.393**	**0.483**
Target	DRNN	0.265	0.294	**0.401**	**0.515**	**0.319**	0.375
	GRU	**0.274**	**0.320**	0.355	0.472	0.309	**0.381**

Table 2: Results for models from grid search over λ, dropout rate and null-class weight. All models have 100-dimensional hidden units and use the GloVe 300-dimensional Common Crawl embeddings.

The same relative performance emerges when more hidden units are added, suggesting the GRU architecture does not see marginal improvement relative to the DRNN architecture as hidden units are added. F1 scores generally improved by about 0.10.

5.3 Qualitative comparisons

The following is an example tagging using the selected models (detailed in 5.1) with 25 hidden units:

DRNN: "[The quickest defeat of Tsvangirai and his $[\text{MDC}]_{target}$ lot would come if they $[\text{chose the path of violence}]_{target}]_{attitude}$," $[\text{an analyst}]_{agent}$ who $[\text{spoke}]_{target}$ on condition of anonymity said.

GRU: "The [$[\text{quickest defeat of Tsvangirai and his MDC lot}]_{target}$ would come if they chose the path $\text{of}]_{attitude}$ violence," [$[\text{an}]_{target}$ $\text{analyst}]_{agent}$ who spoke on condition of anonymity said.

Human: "[The quickest [defeat of Tsvangirai and his MDC lot]$_{target}$ would come if they chose the path of violence]$_{attitude}$," [an analyst]$_{agent}$ who spoke on condition of anonymity said.

Manual inspection of the models' predictions reveals that both models in some cases performed better than the metrics suggest. Most errors are relatively minor albeit inexplicable (such as the GRU model labelling the token "an" as a target on an island). Additional examples reveal that many of the models' false positives in target extraction come from labelling small sequences as targets.

Inspection of the norms of the update matrices (the Wzs) for the GRU nets suggest that the GRU does in fact enable each layer to learn at a different time scale. The norms tended to start out around 10 at the input layer and decrease by an order of magnitude at each successive layer.

6 Conclusion

We have explored the fitness of two types of deep bidirectional recurrent neural networks, one using GRU units in the hidden layers and one using ReLU units in the hidden layers.

We were underwhelmed by the degree to which the GRU architecture outperformed the ReLU architecture. Neither model managed to outperform reported scores for state-of-the-art conditional random field models in the target extraction task (Yang and Cardie 2013), although both beat reported scores in the agent extraction task by a wide margin (Yang and Cardie 2013).

In the future, we may explore adding global reset gates between layers as in (Chung et al. 2015) or performing additional pretraining on the word vectors.

References

Bengio, Yoshua (2009). "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1, pp. 1–127.

Cho, Kyunghyun et al. (2014). "Learning phrase representations using rnn encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078*.

Choi, Yejin et al. (2005). "Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 355–362. DOI: 10.3115/1220575.1220620. URL: http://dx.doi.org/10.3115/1220575.1220620.

Chung, Junyoung et al. (2015). "Gated Feedback Recurrent Neural Networks". In: *CoRR* abs/1502.02367. URL: http://arxiv.org/abs/1502.02367.

El Hihi, Salah and Yoshua Bengio (1995). "Hierarchical Recurrent Neural Networks for Long-Term Dependencies." In: *NIPS*. Citeseer, pp. 493–499.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.

Hinton, Geoffrey E. et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580. URL: http://arxiv.org/abs/1207.0580.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Irsoy, Ozan and Claire Cardie (2014). "Opinion mining with deep recurrent neural networks". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 720–728.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12, pp. 1532–1543.

Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional recurrent neural networks". In: *Signal Processing, IEEE Transactions on* 45.11, pp. 2673–2681.

Wiebe, Janyce, Theresa Wilson, and Claire Cardie (2005). "Annotating expressions of opinions and emotions in language". In: *Language resources and evaluation* 39.2-3, pp. 165–210.

Yang, Bishan and Claire Cardie (2013). "Joint Inference for Fine-grained Opinion Extraction." In: *ACL (1)*, pp. 1640–1649.