Explorations in Identifying and Summarizing Subjective Content in Text

Poorna Kumar
Department of Statistics
Stanford University
poornak@stanford.edu

Viswajith Venugopal
Department of Computer Science
Stanford University
viswa@stanford.edu

Abstract

The extraction and summarization of opinions in text is a useful task that lends itself naturally to connectionist models of sequential data analysis, such as RNNs and LSTMs. Recent work by Irsoy et al. [11] has shown that RNNs (particularly bidirectional deep RNNs) can be quite successfully applied to the task of identifying opinionated phrases in text. Further, Wang et al. [19] have worked on the problem of generating abstractive summaries of opinionated text using an attention-based model and an LSTM neural network. In this work, we explore both these tasks by training various models to perform them, thereby advancing our understanding of the implementation, advantages and challenges of the current approaches to identifying and summarizing opinionated text.

1 Introduction

Given the abundance of opinionated text that we write and read, understanding such text is an important task in natural language processing. In this project, we concern ourselves with two tasks associated with understanding opinions. The first is the identification of subjective phrases in text, and the second is the abstractive summarization of opinionated text. Deep learning methods, particularly those that enable us to model sequential dependencies in text, such as RNNs and LSTMs, are the current state-of-the-art for these problems.

The identification of opinionated phrases in texts is a useful first step in fine-grained opinion analysis, and can inform other tasks such as opinion-oriented question answering and opinion summarizing. We approach this task using bidirectional deep LSTMs, and compare different architectures in terms of their performance. The task is framed as a sequence labelling problem, where the aim is to label every word in the input text according to the presence and type of subjective meaning in the word.

Our second task is the abstractive summarization of opinionated texts. Unlike extractive summarization, which simply selects phrases or sentences from the original text to be included in the summary, abstractive summarization methods can generate text beyond the original input, and have the potential to be more coherent and concise than extractive summaries. We investigate the use of LSTMs with attendion decoders in generating abstractive summaries of texts.

2 Related Work

Early work on understanding opinions in text focused on sentiment analysis at the document level (for example, to distinguish editorials from news, or classify reviews as either positive or negative), or at the sentence level – for example, to classify sentences as subjective and objective, as in [5]. However, it was not uncommon to find two or more opinions in the same sentence, suggesting a need for more granularity; and often, knowing which phrases in a sentence communicated opinion was of interest when building information extraction systems.

In 2005, a new corpus annotation scheme by Wiebe et al. [20] was introduced, that classified words in a corpus based on the subjective meaning they conveyed. In this scheme, words are annotated according to their subjective meaning, by taking into account their context; these annotations are therefore well-suited to bidirectional sequential neural network models, such as bidirectional RNNs (introduced by Schuster and Paliwal in [15]), where the present state depends on both the current word, as well as past and future words (i.e., the context). Hierarchical versions of RNNs, or 'deep' RNNs, are another variant of RNNs, an early description of which is available in [7]. For the problem of annotating text with labels as in [20], Irsoy et al. [11] proposed a learning architecture that consisted of bidirectional deep RNNs, which beat the existing state-of-the-art semi-CRF model employed by Yang and Cardie in [21].

A related problem in opinion-understanding is to generate short summaries of opinionated texts, so as to be able to extract salient information from, say, sets of reviews, or from points in a debate. Early work on automatic summary generation focuses on extractive summarization, such as Hu and Liu's work [10] in which product reviews were summarized by identifying which salient opinionated sentences to include in the summary. The drawback of this approach is that the generated summaries invariably include secondary or irrelevant information.

On the other hand, abstractive summarization techniques can include text that isn't in the original input, giving them the power to be more coherent and precise. Most existing approaches to this are based on identifying salient phrases, and then combining them into sentences, such as the work of [3]. However, this system is not capable of generating new words and suffers from ungrammatical sentence structure. In contrast, the approach by Gerani et al. [8] uses a large amount of human input to come up with templates of summary sentences, that are more likely to be grammatical. In 2016, Wang et al. came up with a neural-network based abstract generation system [19] for summarizing text, that was shown to produce more grammatical sentences and did not suffer from the drawback of requiring large amounts of human input. Their approach was to treat the problem similarly to the machine translation problem, with an encoder and decoder structure, implemented using deep bidirectional LSTMs and attention. This is the current state-of-the-art, and we follow this approach.

3 Approach and Methods

3.1 Opinion Identification

In this section, we give a brief overview of the architectures we used for the opinion identification problem.

3.1.1 Recurrent Neural Networks (RNNs)

A recurrent neural network (RNN)[13] is a versatile model which is useful for modelling sequences. At every step, the RNN has a hidden state. The current hidden state is computed based on the current input and the previous hidden state, and the current output is computed from the current hidden state. Crucially, the same parameters are used at every time step. The following equations describe a vanilla RNN with input x_t, hidden state h_t and output y_t at time t (excluding the biases):

$$h_t = f_H(W^{hh}h_{t-1} + W^{hx}x_t) \tag{1}$$

$$y_t = f_O(W^{hy}h_t) \tag{2}$$

where f_H and f_O are non-linearities like sigmoid, tanh or ReLu.

3.1.2 Long Short Term Memory (LSTMs)

Long Short Term Memory (LSTM) [9] is a class of models which has the same recurrent structure as an RNN, but a more complex computation in each cell. In particular, LSTMs have input, forget and output gates which control how much of the input is used, how much is forgotten, and how much is output respectively. The main advantage of an LSTM over an RNN is that it can retain information over more time steps, and is less affected by the vanishing gradient problem. We do not reproduce the equations for lack of space; the unacquainted reader is referred to [6] for an insightful explanation.

3.1.3 Deep (Stacked) RNNs and LSTMs

The vanilla RNNs and LSTMs described above can be enhanced by *stacking* cells into multiple layers. [7] Each layer works exactly like before: however, now, the output of a cell is fed in as the input to the cell above it, and the outputs from the top layer are taken as the model's outputs. The idea behind stacking is that higher layers can possibly model higher-order dependencies in the model – that is, different layers can function at different time scales.

3.1.4 Bidirectional RNNs and LSTMs

For certain sequence modelling problems, it might be the case that the output is better determined after considering not only past, but also *future* inputs. Bidirectional RNNs and LSTMs help with this.[15] A bidirectional RNN can be formulated as follows:

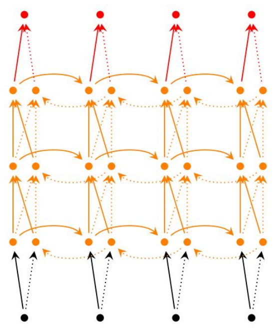
$$\overrightarrow{h}_t = f_H(\overrightarrow{W} x_t + \overrightarrow{V}\,\overrightarrow{h}_{t-1} + \overrightarrow{b}) \tag{3}$$

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

$$y_t = f_O(\overrightarrow{U}\,\overrightarrow{h_t} + \overleftarrow{U}\,\overleftarrow{h_t} + c) \tag{5}$$

where variables with $\rightarrow$ represent computation in the forward direction, and those with $\leftarrow$ represent computation in the reverse direction. We have a different set of weights for each direction to compute one hidden state for each direction, and we use both of them to compute the output (note that the above formulation essentially concatenates the vectors). Figure 1 is a visualization of a stacked, bidirectional RNN.

Figure 1: A visualization of a deep bidirectional RNN.



3.2 Opinion Summarization

In this section, we give a brief overview of the architectures we used for the opinion summarization problem.

3.2.1 Sequence to Sequence Models

Sequence to sequence models[17] go over the entirety of an input sequence, and then output the predicted sequence. They are used, for example, in translation and summarization. The RNN-based models described above can be adapted to perform sequence to sequence tasks as well: in this case, there is an encoding phase, where the model goes over the input sequence but does not output anything, and a decoding phase, where the model, starting from the final encoded hidden state, outputs something at every time step. At train time, during decoding, the model is fed the target sequence as input, and is trained to predict the next word. At test time, the model is fed, at every stage, its output from the previous time step.

3.2.2 Attention Decoding

The weakness of sequence to sequence models is that the final hidden state of the encoder needs to capture the entire input sequence, which makes it difficult for long sequences. Attention [2] is a mechanism used to handle this problem. Here, when at time step t of the decode phase, the entire input sequence is fed to the model as a weighted sum over the encoder hidden states, $s_t = \sum_i a_i^{(t)} h_i$. $a_i^{(t)}$ is the attention coefficients at time t for input i. It is computed with a softmax over all input:

$$a_i^{(t)} = \text{softmax}(W_s \cdot tanh(W_{cg} h_i + W_{hg} b_{t-1})) \tag{6}$$

where b_{t-1} is the previous hidden state of the decoder, and the w matrices are parameters to be learned. Intuitively, we want this function to compute how likely to input word i is to be used to generate the next word in the summary.

3.3 Additional Features of Models

3.3.1 Regularization with Dropout

Dropout [16] is a simple method to prevent neural networks from overfitting, where we randomly set some of the values to zero during the forward pass of our neural network. Dropout is applied to each value independently, with the same probability. Dropout acts as a kind of regularizer, and has been shown to have many of the benefits of model ensembling.

3.3.2 Optimization with Adam

Adam [12] is a stochastic gradient descent optimization algorithm which works very well in practice, and which, over the course of our experiments, we found to work better than vanilla Gradient Descent. Concretely, the update from x_t to x_{t+1} is done using the gradient dx and the learning rate γ as follows:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) dx \tag{7}$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) dx^2 \tag{8}$$

$$x_{t+1} = x_t - \gamma \frac{m_{t+1}}{\sqrt{v_{t+1} + \epsilon}} \tag{9}$$

Here, β_1 and β_2 are decay parameters, m is a momentum vector (which can be interpreted as the "smoothened" gradient), and v is a vector that enables our updates to be per-parameter adaptive. (A more detailed explanation can be found at [18]).

4 Experiments

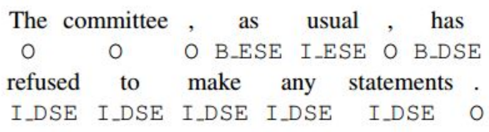
In this section, we talk about our datasets, evaluation metrics and the experiments we ran.

4.1 Datasets

4.1.1 Opinion Identification

The dataset used for this task is the MPQA 1.2 corpus (535 news articles, 11,111 sentences) [20] which contains news articles on ten different topics, annotated as in [20]. Each word in the corpus has been manually annotated, and in particular, we are interested in whether it has been labeled as part of an expressive subjective expression (ESE), which indirectly conveys the opinion-holder's attitude, or a direct subjective expression (DSE), which directly conveys the opinion-holder's attitude. Additionally, if a word is a part of a DSE phrase, it is labeled as either being in the beginning (BDSE) or inside (IDSE) of the phrase, and similarly with ESE phrases. Thus, every word in our dataset is labeled as either BDSE, IDSE, BESE, IESE or O, where O indicates that the word does not communicate an opinion. An example labeled sentence from our corpus can be found in Figure 2.

Figure 2: An example labelled sentence from the dataset.

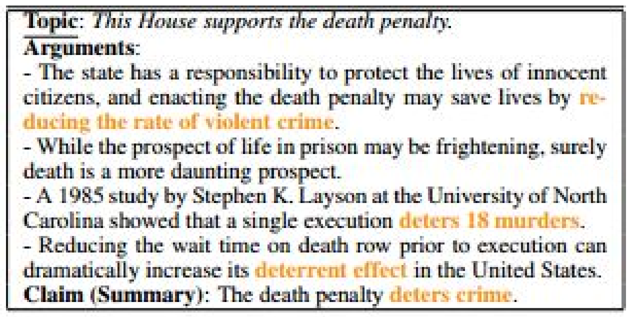


The	committee	,	as	usual	,	has
O	O	O	B_ESE	I_ESE	O	B_DSE
refused	to	make	any	statements	.	
I_DSE	I_DSE	I_DSE	I_DSE	I_DSE	O	

4.1.2 Opinion Summarization

We use an argumentation dataset from idebate.org, a Wikipedia-style website for gathering pro and con arguments on controversial issues. The arguments under each debate are organized into different ’for’ and ’against’ points, and each point contains a one-sentence central claim written by an editor which summarizes the corresponding arguments. This is treated as the gold-standard summary. An example from our dataset is shown in Figure 3.

Figure 3: An example set of arguments, along with their summary claim.



Topic: *This House supports the death penalty.*
Arguments:
- The state has a responsibility to protect the lives of innocent citizens, and enacting the death penalty may save lives by reducing the rate of violent crime.
- While the prospect of life in prison may be frightening, surely death is a more daunting prospect.
- A 1985 study by Stephen K. Layson at the University of North Carolina showed that a single execution deters 18 murders.
- Reducing the wait time on death row prior to execution can dramatically increase its deterrent effect in the United States.

Claim (Summary): The death penalty deters crime.

4.2 Evaluation Metrics

4.2.1 Opinion Identification

To select and evaluate our model, we measure the success of our model on validation (and later test) data. The evaluation metric we use is based on the degree of overlap between our identified opinionated phrases and the true opinionated phrases in the text. We compute two different measures of overlap – binary and proportional overlap. In binary overlap (the kinder measure), we count a true opinionated phrase to be correctly identified if it has any overlap with a predicted opinionated phrase with the same label. In proportional overlap, we take the degree of overlap into account, measuring accuracy as a fraction of overlap between phrases. For both binary and proportional overlap, we can calculate precision, recall and F1 scores.

4.2.2 Opinion Summarization

We evaluate our model by perplexity scores on the validation and test sets. If $\hat{y}_i$ is the output of the final softmax layer for the i^{th} word in our summary, W_s is the number of words in the summary s, and y_i is the desired vector (it is one-hot, so without loss of generality, let $y_{ik_i} = 1$). Then, the soft-max cross-entropy loss for each summary s generated is $J_s = \sum_{i=1}^{W_s} \log(\hat{y}_{ik_i})$.

Now, if we sum over all losses J_s, we get a total softmax-CE loss as $J = \sum_s J_s$. Our perplexity is then given by $P = \exp(J)$.

4.3 Overview of Implementation

We used TensorFlow[1] [1], an open source library developed by Google to implement our models. To feed our data in to our models, we tokenized our inputs (for which we used NLTK [4]), and com-

[1]All our code can be found at `https://github.com/viswajithiii/cs224d-project`

Table 1: F1 scores of the RNN opinion tagger with the ReLu non-linearity for various depths

Depth	Hidden Units	DSE		ESE	
		Binary F1	Proportional F1	Binary F1	Proportional F1
5	46	0.5812	0.5153	0.5754	0.4569
4	51	0.6052	0.5513	0.6042	0.4866
3	58	0.5992	0.5393	0.598	0.4889
2	70	0.6084	0.5553	0.6113	0.4975
1	94	0.5846	0.5295	0.5858	0.4693

puted word embeddings using pre-trained word vectors from GloVe [14] (300-dimensional vectors trained on the 840B token CommonCrawl dataset) to encode our input tokens for both the opinion identification and the opinion summarization task. We chose to update our word vectors for the latter, but left them frozen for the former.

To handle inputs sequences of variable length while training our network for the summarization task, we used Tensorflow's implementation of bucketing in its seq2seq library. Bucketing is a method wherein input sequences of similar lengths are grouped in buckets. Within a bucket, each sequence is padded with a special PAD symbol to bring it to a uniform length (the length of the longest sequence in the bucket). Then, the sequences can be fed into the network in batches. This helps us avoid having to build a new computation graph for each sequence of a different length, but allows us the flexibility to handle variable length sequences without all the padding that would be required without buckets.

For all our models, we found that the Adam optimizer worked best, with a learning rate of 0.005 for the opinion identification problem, and 0.001 for the summarization problem.

4.4 Opinion Tagging

For opinion tagging, we used bidirectional RNNs with various architectures. The following sections detail our experimental findings. Overall, all our models find it easier to identify DSEs than ESEs, which is unsurprising, since ESEs are more subtle.

4.4.1 RNN vs LSTM

We performed experiments using both bidirectional RNNs (with ReLU nonlinearity) and LSTMs (with tanh non-linearity) of varying depth, and we summarize our results in Tables 1 and 2. LSTMs perform better than RNNs on this task, which is probably because identifying opinions will require the model to understand long-term dependencies.

4.4.2 The Benefits of Stacking

To investigate the effect of depth on performance, we used LSTMs of different depths and then compared their performances on the dev set. In order to be able to compare more effectively, we held the total number of parameters across architectures approximately equal (around 200,000) – that is, networks got narrower (with fewer hidden units) as they got deeper (with more stacked layers). Our results in Table 2 indicate that adding depth is (somewhat) beneficial upto 3 layers, after which the returns of adding depth degrade.

4.4.3 Dropout Regularization

We performed the opinion tagging task using LSTMs both with and without dropout, and a comparison of performance is given in Table 2. Our conclusion is that dropout (with a 'keep' probability of 0.9) usually helps improve F1 scores on the validation set. Similarly, our RNN experiments all employed dropout, with a 'keep' probability of 0.9.

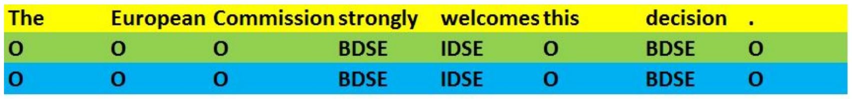
Table 2: F1 scores of the LSTM opinion tagger for various depths

Depth	Hidden units	Dropout	DSE		ESE	
			Binary F1	Proportional F1	Binary F1	Proportional F1
5	46	No	0.608	0.546	0.564	0.450
5	46	Yes	0.635	0.576	0.592	0.485
4	51	No	0.643	0.585	0.579	0.477
4	51	Yes	0.618	0.564	0.591	0.483
3	58	No	0.609	0.554	0.602	0.492
3	58	Yes	0.638	0.579	0.592	0.482
2	70	No	0.631	0.584	0.570	0.467
2	70	Yes	0.646	0.596	0.617	0.512
1	94	No	0.599	0.555	0.599	0.492
1	94	Yes	0.608	0.548	0.589	0.481

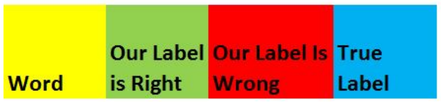
4.4.4 Qualitative Results: Performance and Error Analysis

For the opinion tagging task, our models perform quite well. The following examples demonstrate some successes in identifying DSEs and ESEs, with our LSTM of depth 3 that employs regularization.

Figure 4: Correct DSE Examples

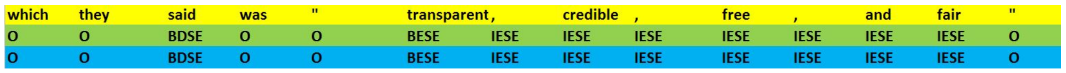


The	European	Commission	strongly	welcomes	this	decision	.
O	O	O	BDSE	IDSE	O	BDSE	O
O	O	O	BDSE	IDSE	O	BDSE	O



Word	Our Label is Right	Our Label Is Wrong	True Label

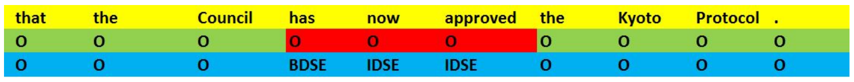
Figure 5: Correct ESE Examples



which	they	said	was	"	transparent	,	credible	,	free	,	and	fair	"
O	O	BDSE	O	O	BESE	IESE	IESE	IESE	IESE	IESE	IESE	IESE	O
O	O	BDSE	O	O	BESE	IESE	IESE	IESE	IESE	IESE	IESE	IESE	O

Some of its more notable misses are below. Figure 6 shows an instance when our network misses a valid DSE.

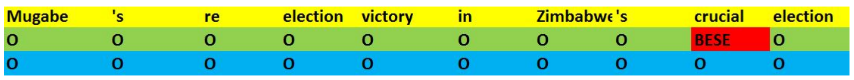
Figure 6: Wrong DSE Examples



that	the	Council	has	now	approved	the	Kyoto	Protocol	.
O	O	O	O	O	O	O	O	O	O
O	O	O	BDSE	IDSE	IDSE	O	O	O	O

Figure 7 shows how our network mislabels a word as an ESE, when in fact it hasn't been labeled as one. On closer inspection, though, it appears as though an ESE is a reasonable guess for the word.

Figure 7: Wrong ESE Examples



Mugabe	's	re	election	victory	in	Zimbabwe	's	crucial	election
O	O	O	O	O	O	O	O	BESE	O
O	O	O	O	O	O	O	O	O	O

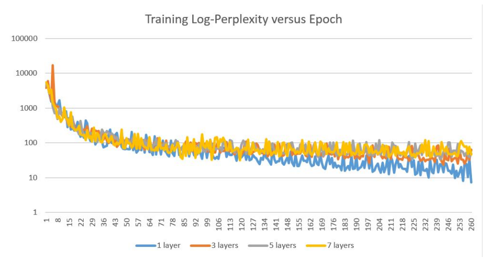
4.5 Opinion Summarization

For the opinion summarization, we used sequence to sequence LSTMs with attention over the inputs while decoding.

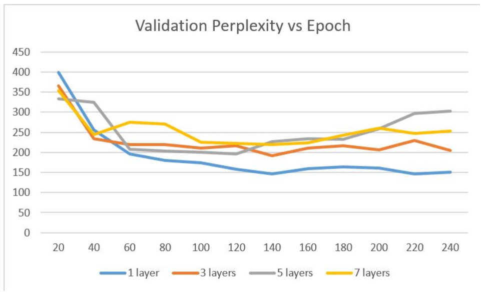
4.5.1 Perplexity under different architectures

First, we tried different architectures with various levels of model stacking. The training and validation perplexity curves versus epoch are shown in Figures 8a and 8b. It is surprising to note that the model with one layer performs best in terms of perplexity; this is likely because the sequence to sequence task requires more data in order to train the model with more layers.

(a) Training Log Perplexity Versus Epoch



(b) Validation Perplexity Versus Epoch



4.5.2 Bootstrapping weights by training it to echo input

When we first bootstrapped weights by training the network to echo its own input, we found that the quality of our output summaries increased. This could be explained by the fact that the optimization problem of a neural network is highly non-convex, and hence the convergent weights of the network are highly dependent on the starting point. Choosing a good starting point, increases the chances that the network will converge to a good set of final weights. The best validation perplexity we achieved without this starting point was 160, while with this starting point we were able to reach a validation perplexity of 30.

4.5.3 Qualitative Results: Performance and Error Analysis

Although our model does well in terms of perplexity, our outputs are not as spectacular as we would like them to be. Below are some example outputs:

- **Input:** It is sufficient for the decriminalization of sadomasochism that each participant is aware of the hazards inherent in the fetishes they will be exploring and consents to them. **Output:** *The suffering is the harm a a to . the right to enable the right of to the right way*
- **Input: The rich help the poor only to make themselves feel better.**
  Output: *It is the economy the EU*

We can see that our model is not perfect in terms of grammar, but it appears to grasp the general topic. Further, in these examples, it is able to generate words that are not in the input text in its output, which is the idea behind abstractive summarization.

5 Conclusion and Future Steps

While our opinion tagging network achieves good results (in fact, sometimes it picks up things that haven't been manually annotated as opinionated, but probably *should* have been), our summarization system could be significantly improved by using a larger dataset, and more hand-engineered features.

In the future, for the summarization task, we intend to implement the importance sampling scheme described in [19] to subsample the input sentences based on their relative importance, as determined by various features such as the number of words in them, their POS tags, named entities, etc. This will ensure that we feed in more relevant inputs to the encoder. We could also employ a ranking system, wherein the k best abstracts generationed are ranked by heuristics, as suggested in [19].

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] L. Bing, P. Li, Y. Liao, W. Lam, W. Guo, and R. J. Passonneau. Abstractive multi-document summarization via phrase selection and. *arXiv preprint arXiv:1506.01597*, 2015.

[4] E. L. Bird, Steven and E. Klein. Natural language processing with python. 2009.

[5] R. F. Bruce and J. M. Wiebe. Recognizing subjectivity: a case study in manual tagging. *Natural Language Engineering*, 5(02):187–205, 1999.

[6] Colah. Understanding lstm networks. http://colah.github.io/posts/2015-08-understanding-lstms/.

[7] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. Citeseer.

[8] S. Gerani, Y. Mehdad, G. Carenini, R. T. Ng, and B. Nejat. Abstractive summarization of product reviews using discourse structure. Citeseer.

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

[10] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[11] O. Irsoy and C. Cardie. Opinion mining with deep recurrent neural networks.

[12] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] L. Medsker and L. Jain. Recurrent neural networks. *Design and Applications*, 2001.

[14] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[15] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[18] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[19] L. Wang and W. Ling. Neural network-based abstract generation for opinions and arguments.

[20] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.

[21] B. Yang and C. Cardie. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. Association for Computational Linguistics, 2012.