Document-Level Entity-to-Entity Sentiment Analysis with LSTM-Based Models

Belinda Li
Paul G. Allen School of Computer Science & Engineering
Univ. of Washington, Seattle, WA

Abstract

Most work in fine-grained sentiment analysis is focused on predicting sentiment from within a sentence. These sentences usually involve only a single sentiment being expressed from one holder to one target. Instead, we aim to perform document-level entity-to-entity sentiment analysis, whereby we predict various sentiments among all entities at the document level. These sentiments may be expressed within or across sentences, and different entities may hold different sentiments towards each other. Sentence-level models which predict a single sentiment do not generalize well to our task, as they are unable to account for cross-sentence sentiments. Therefore, we propose two novel LSTM-based architectures for this task, and show that both models outperform sentence-level baselines. Surprisingly, however, we find that our models do not outperform state-of-the-art SVM-based models. In particular, we find that LSTM-based models have difficulty classifying entities which express no sentiment towards each other, a limitation which accounts for 60-80% of its mistakes on the development set.

1 Introduction

Historically, research in sentiment analysis has focused on finding the overall sentiment of a short span of text, like tweets (Mohammad et al., 2013; Tasoulis et al., 2018; Küçük and Can, 2018), product reviews (Fang and Zhan, 2015), or movie reviews (Pang and Lee, 2004). However, in recent years, there has been a shift towards more fine-grained opinion mining. Not only is the overall sentiment of a piece of text determined, but the opinion holder and target are also identified. This allows for a much more nuanced understanding of the sentiments expressed within a text.

Much of the work in fine-grained sentiment analysis has focused on classifying sentiments within a single sentence, containing a single entity pair mention. However, sentiments are expressed in all types of texts, many of which contain multiple entity pairs and multiple mentions of each entity. News articles in particular often encode complex networks of opinions amongst a multitude of entities. Many of these sentiments are expressed across sentences and cannot be captured by sentence-level models. Take this article from Xinhua News Agency as an example:

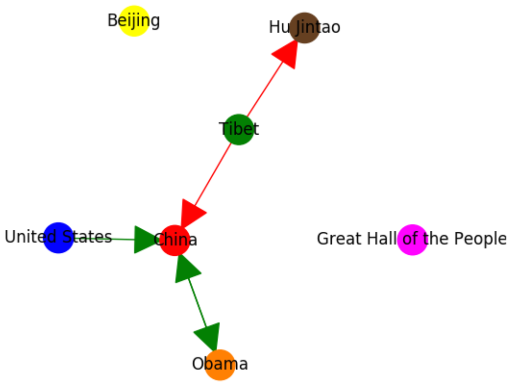

Figure 1: Graph showing the network of sentiments expressed amongst various entities in the Xinhua News Agency article. Arrows are directed from opinion holder to opinion target. Red arrows indicate negative sentiment and green arrows indicate positive sentiment. Disconnected nodes represent entities that are present in the text, but do not participate in any sentiment relations, either as opinion holders or opinion targets.

> URGENT: **Obama** says **U.S.** recognizes **Tibet as part of China**. **United States** President **Barack Obama** Tuesday said the **U.S.** government recognizes that **Tibet** is **part of the People's Republic of China**. He also said that **the United States** supports the early resumption of dialogue between the **Chinese** government and representatives of

> the Dalai Lama to resolve any concerns and differences that the two sides may have. "**The United States** respects the sovereignty and territorial integrity of **China**," **Obama** said at a joint press conference with **Chinese** President **Hu Jintao** at **Beijing**'s **Great Hall of the People**.

Five sentiments have been labelled within this article:

1. **Obama** is positive towards **China**
2. **United States** is positive towards **China**
3. **China** is positive towards **Obama**
4. **Tibet** is negative towards **China**
5. **Tibet** is negative towards **Hu Jintao**

The fifth sentiment, in particular, involves entities that are never even mentioned in the same sentence, requiring cross-sentence reasoning.

We formulate our task of entity-to-entity sentiment analysis in a pipelined manner. In the first step, all entities within a document are extracted and labelled. Co-reference resolution is performed to identify all mentions of each entity. In the second step, we consider each pair of entities and classify their sentiment relationship into three possible types: positive, negative, or none. This is opposed to previous approaches which first identified the sentiment of a text, and then the holder and target of that sentiment (Kim and Hovy, 2006). Such an approach works well for short texts expressing a single main sentiment, but works less well for long documents containing multiple sentiment relations. As strong NER and coreference (Lee et al., 2017) models exist already, we focus the bulk of our work on the second task, allocating the first task to preprocessing. We propose two novel neural architectures for this task, and show that it outperforms sentence-level baselines.

2 Data

We use five datasets from Choi et al. (2016) for training, development, and testing. These datasets contain the text of a document, and annotations for each entity pair in the document. The training set ("train-original") comprises of news articles from the Gigaword corpus[1]. Human-annotated labels were generated for this set via crowdsourcing. The KBP set, derived from the KBP competition, was used for development and testing. Gold-standard annotations for this set were supplemented with human annotations to create labels between all entity pairs. The KBP set was then split, with 50% of the documents being used for development and 50% for testing ("test-KBP"). The development set itself was also split, as 50% of the development data was used for hyper-parameter tuning ("dev-tune") and the rest for evaluation ("dev-eval"). A second test set ("test-MPQA") was also used, derived from densely-labelled MPQA dataset.

2.1 Training Data

The distribution of the original training set does not mimic that of the development and test sets, most notably due to its lack of "none" labels. Such domain differences can problematic for deep neural networks in NLP (Mou et al., 2016). Therefore, to optimize our model's performance, we make two changes to the training set. First, we incorporate dev-tune into our training set, using dev-eval for both hyper-parameter tuning and evaluation. As dev-tune, dev-eval, and test-KBP are all derived from the same corpus, their distributions should match. Second, we supplement the training data with weakly generated "none" pairs. As the original training set did not have annotations between all pairs of entities within a document (whereas all the other dev and test sets did), we can generate new examples by finding un-labelled pairs and assuming those pairs hold no sentiment. We then randomly select 10% of the newly generated pairs and add them to the training data, which was sufficient to increase the proportion of "none" pairs in the training data to 85%, comparable to the 80-85% found in the development and test sets. Table 1 shows that the distribution of the new training dataset ("train-new") is now much closer to that of the development and test sets.

2.2 Preprocessing

We use preprocessed data from Choi et al. (2016), discarding part-of-speech tags and dependency paths which are not used by our model. Instead, the preprocessing pipeline for our model consists of only three steps: tokenization, NER, and coreference resolution. Stanford CoreNLP (Manning et al., 2014) was used for all three steps. Following NER, entities of type date, duration, money, time, and number were discarded, as they

[1] LDC2014E13:TAC2014KBP English Corpus

Dataset	Docs	Entities / Doc	Neg	None	Pos	% None
train-original	897	2.63	648	815	355	44.8
train-new	949	8.98	973	11306	1013	85.1
dev-tune	38	8.82	158	2349	365	81.8
dev-eval	37	9.08	174	2454	404	80.9
test-KBP	79	9.25	437	5459	718	82.5
test-MPQA	54	11.72	521	6464	625	84.9

Table 1: Dataset statistics. For each dataset, shows the total number of articles, the average number of entities within each article, the total number of negative pairs, none pairs, positive pairs, and the percentage of all pairs that are none.

do not tend to participate in sentiment relations. Coreference resolution was performed on the remaining entities to isolate all mentions of the entity.

3 Models

We formulate our task by considering all entities within a document as potential holders and targets of sentiments. If there are E entities in the document, then we consider $E(E-1)$ holder-target pairs for the document. For each pair, we classify its sentiment into one of {positive, negative, none}. We explore two different neural models for this task.

3.1 Attentive BiLSTM ("ABL")

Inputs. We concatenate three sets of embeddings to represent each token x_t: word embeddings (w_t), learned polarity embeddings (p_t), and learned holder-target embeddings (e_t):

$$x_t = [w_t, p_t, e_t]$$

Pre-trained GloVe vectors (Pennington et al., 2014) are used for word embeddings. We also explicitly encode the polarity of known subjectivity clues (words which may be used in a positive or negative context) by appending embeddings for its polarity. We use a pre-existing lexicon of over 8000 subjectivity clues from Wilson et al. (2005) and learn different embeddings for positive, negative, and neutral words. Finally, our model architecture has no notion of which tokens are holders and which are targets as it treats all tokens the same. Thus, we rely on a holder-target embedding to encode whether each token is part of a holder or target span.

LSTM. We feed each embedded token x_t into a bidirectional LSTM to encode it within its context.

$$\begin{aligned}
f_{t,\delta} &= \sigma(\mathbf{W}_f[x_t, h_{t+\delta,\delta}] + b_i)\\
o_{t,\delta} &= \sigma(\mathbf{W}_o[x_t, h_{t+\delta,\delta}] + b_o)\\
\tilde{c}_{t,\delta} &= \tanh(\mathbf{W}_c[x_t, h_{t+\delta,\delta}] + b_c)\\
c_{t,\delta} &= f_{t,\delta} \circ \tilde{c}_{t,\delta} + (1 - f_{t,\delta}) \circ c_{t+\delta,\delta}\\
h_{t,\delta} &= o_{t,\delta} \circ \tanh(c_{t,\delta})\\
x^*_t &= [h_{t,1}, h_{t,-1}]
\end{aligned}$$

whereby $\delta = \{1, -1\}$ indicates directionality of the LSTM. The final representation of the token x^*_t is a concatenation of the output from the forward LSTM and that from the backward LSTM.

Attention. We use an attentional mechanism over all contextually-encoded tokens $x^*_t \in \{x^*_0, \ldots, x^*_T\}$ to extract a final sentiment score $s_{i,j}$ from holder i to target j.

$$\alpha_t = \mathbf{W} \cdot x^*_t$$

$$a_{i,j,t} = \frac{\exp(\alpha_t)}{\sum_{k=0}^{T} \exp(\alpha_k)}$$

$$s_{i,j} = \sum_{t=0}^{T} a_{i,j,t} \cdot x^*_t$$

Hyperparameters. The GloVe embeddings, learned polarity embeddings, and learned holder-target embeddings are 50-dimensional each. For the biLSTM, we use 2 stacked layers and a hidden dimension of 50. We apply a dropout to the hidden layer of the LSTM with probability 0.2. For training, we optimize the model using Adam with α = 1e-3, β_1 = 0.9, β_2 = 0.999, ϵ = 1e-8. We use a batch size of 50, training up to 15 epochs, with epoch 6 being selected as the optimal epoch based on development set performance.

3.2 Pairwise Attentive BiLSTM ("PABL")

We introduce a novel architecture for the classification of opinion sentiment. This model is built

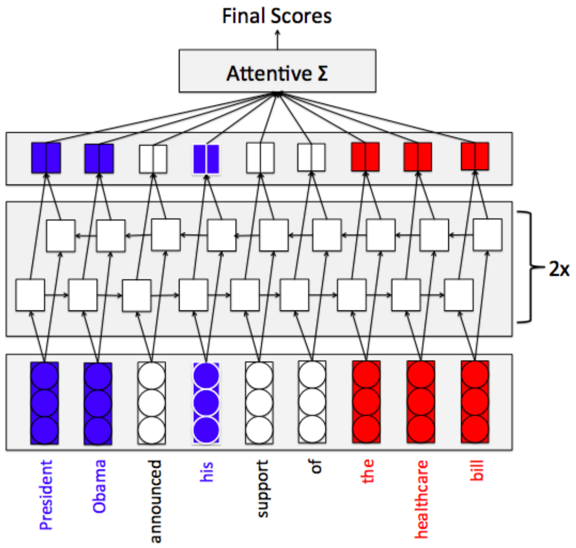

Figure 2: ABL architecture. Embedded tokens are passed into a 2-layer bidirectional LSTM. An attentive sum is performed over the outputs to generate final scores. In this example, blue is used to represent the holder, while red is used to represent the target. Sample text is only one sentence, but actual documents are much longer.

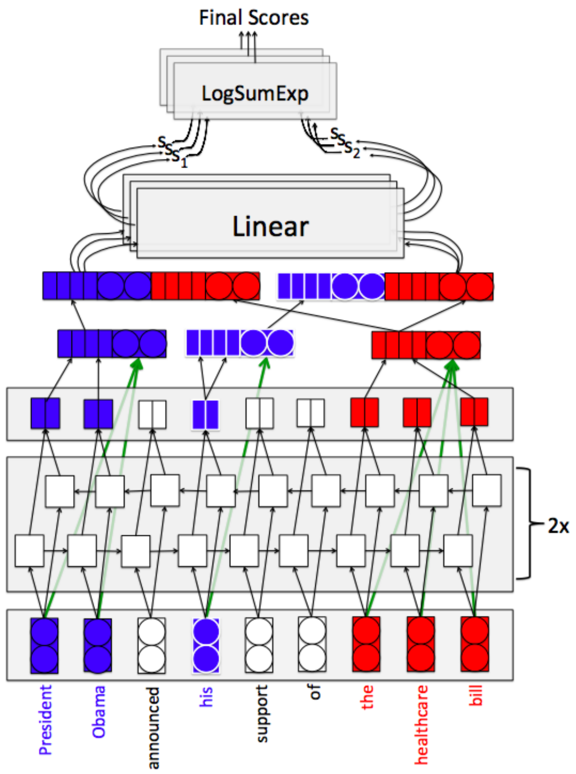

Figure 3: PABL architecture. Span representations are generated by concatenating encoded start and end tokens with an attention mechanisms over embedded tokens in the span. Next, each holder span and target span are concatenated to form pairwise representations, which are fed through a linear operator to generate pairwise mention scores. Finally, mention scores are aggregated using LogSumExp.

off of ABL, with additional connections added on top. We borrow components of this model from state-of-the-art models in relation extraction (Verga et al., 2018) and co-reference resolution (Lee et al., 2017).

Inputs. We concatenate two sets of embeddings to represent each token x_t: word embeddings (w_t) and polarity embeddings (p_t):

$$x_t = [w_t, p_t]$$

Both embeddings are identical to those used in ABL. Note that we do not need to append holder-target embeddings as the relation is built into the model architecture itself.

LSTM. As in ABL, we use a bidirectional LSTM to contextually encode each embedded token x_t.

$$x_t^* = [h_{t,1}, h_{t,-1}]$$

Span Representations. To represent multi-word mentions and entities ("spans"), we use span representations based on those used by Lee et al. (2017). For each span i, we take the LSTM-encoded tokens in the span and concatenate them in the following manner:

$$\mathbf{g}_i = [x^*_{START(i)}, x^*_{END(i)}, \hat{\mathbf{x}}_i]$$

whereby $\mathbf{x}^*_{START(i)}$ and $\mathbf{x}^*_{END(i)}$ are the endpoints of the span, and $\hat{\mathbf{x}}_t$ is computed using an attention mechanism over each embedded word in the span. Following (Lee et al., 2017)'s work, we compute the attentive representation $\hat{\mathbf{x}}_i$ as a weighted sum of each embedded word x_t in the span. We make a slight modification in that we compute attention directly from the embedded token themselves, rather than deriving them from the LSTM-encoded outputs.

$$\alpha_t = \mathbf{w}_\alpha \cdot FFNN_\alpha(x_t)$$

$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=START(i)}^{END(i)} \exp(\alpha_k)}$$

$$\hat{\mathbf{x}}_i = \sum_{t=START(i)}^{END(i)} a_{i,t} \cdot x_t$$

Pairwise Scoring. Verga et al. (2018) scores pairs of triples using an MLP followed by a bilinear operator. To reduce the number of parameters, we instead use a linear transformation of concatenated holder-target representations. We represent all combinations of the n holder mentions

$\mathbf{g}_1^{holder}, \mathbf{g}_2^{holder}, \ldots, \mathbf{g}_n^{holder}$ and the m target mentions $\mathbf{g}_1^{target}, \mathbf{g}_2^{target}, \ldots, \mathbf{g}_m^{target}$ using concatenation, generating a total of nm pair representations.

$$\mathbf{p}_{i,j} = [\mathbf{g}_i^{holder}, \mathbf{g}_j^{target}, \phi(i,j)]$$

where ϕ represents feature vectors for the pair (i, j).

We then pass each pairwise representation through a linear layer. This generates scores for each (holder mention i, target mention j, sentiment s) triple, where $s \in$ {positive, negative, none}.

$$A_{i,j,s} = \mathbf{M} \cdot \mathbf{p}_{i,j}$$

Features. For each holder-target mention pair, we learn separate embeddings for four different sets of features,

$$\phi(i,j) = [\phi_s(i,j), \phi_c(i,j), \phi_f(i,j), \phi_r(i,j)]$$

which we append to its pairwise representation $\mathbf{p}_{i,j}$. ϕ_s encodes how a sentence baseline model predicted the example, with different embeddings for positive, negative, and no sentiment predictions. ϕ_c is a co-occurrence feature, representing the number of sentences in which the holder and target both appeared. Finally, we have features ϕ_f and ϕ_r encoding the total number of mentions of the holder entity and target entity. Whereas ϕ_f ("frequency") encodes the raw number of mentions of each entity, ϕ_r ("rank") encodes the "rank" of the entity, with the most frequently mentioned entity ranked 1, the second-most frequently mentioned ranked 2, and so on.

Aggregation. Our data contains labels for each entity pair, but not for each mention pair. To deal with this, we perform multi-instance learning following Verga et al. (2018), whereby we train the model over the aggregate of the mention pairs and perform a single update for the aggregate. We use LogSumExp as the aggregation function. This function is a smooth approximation of the max, allowing us to extract the most probable scores for each sentiment s.

$$\begin{aligned} &score(\text{holder, target}, s) \\ &\qquad = \log \sum_{\substack{i \in \{1,\ldots n\},\\ j \in \{1,\ldots m\}}} \exp\left(A_{i,j,s}\right) \end{aligned}$$

Training. We aim to maximize the log-likelihood of the correct labels for each example. We can represent each example by the triple (document D, holder entity h, target entity t).

Let $\mathcal{Y} = \{\text{Positive, Negative, None}\}$ represent the set of all possible labels for each example. Given N training examples labelled $y_1, \ldots, y_n \in \mathcal{Y}$, with document D, the likelihood is given by

$$\begin{aligned} &P(y_1, \ldots, y_n | D, h, t) \\ &\qquad = \prod_{i=1}^{N} P(y_i | D, h, t) \\ &\qquad = \prod_{i=1}^{N} \frac{\exp\left(score(h,t,y_i)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(score(h,t,y')\right)} \end{aligned}$$

We aim to maximize the the log-likelihood

$$\sum_{i=1}^{N} \log \left(\frac{\exp\left(score(h,t,y_i)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(score(h,t,y')\right)} \right)$$

Hyperparameters. The GloVe embeddings and learned polarity embeddings are 50-dimensional each. All features are represent as 25-dimensional learned embeddings. The co-occurrence feature ϕ_c are binned into buckets: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10+]. The mention features ϕ_f and ϕ_r are binned into [1, 2, 3, 4, 5+]. For the biLSTM, we use 2 stacked layers and a hidden dimension of 50. We apply a dropout to the input and all hidden layers of the LSTM with probability 0.2. For training, we optimize the model using Adam with α = 1e-3 and a weight decay of 1e-5. We also set β_1 = 0.9, β_2 = 0.999, ϵ = 1e-8. We use a batch size of 50 and train up to 10 epochs, with epoch 6 being selected as the optimal epoch based on development set performance.

4 Results

To measure our models' performance on our task, we use precision, recall, and F1 scores. In general, our model, which has the capacity to perform cross-sentence reasoning, performs significantly better than sentence-level baselines. For all neural models, recall is much higher than precision if there is a sentiment (i.e. positive/negative label), while precision is much higher than recall if there is no sentiment (i.e. none label). Notably, while our F1 scores on positive and negative sentiment do not exceed that of Choi et al. (2016), our recalls are much higher than theirs. This suggests that neural models tend to more conservative in picking "no sentiment" labels. This additionally suggests that combining aspects of both models may be beneficial.

		Neg			None			Pos		
		P	R	F1	P	R	F1	P	R	F1
dev-eval	Sentence Baseline	11.3	**73.0**	19.6	90.6	64.8	75.6	44.8	17.1	24.7
	Choi et al. (2016) SVM base	25.6	36.8	30.2				47.3	36.9	41.4
	Choi et al. (2016) SVM+ILP	**37.2**	35.1	36.1				**58.2**	37.9	**45.9**
	Attentive biLSTM	11.8	59.7	19.9	90.1	65.8	76.0	32.5	19.9	24.7
	Pairwise Attentive biLSTM	31.9	63.2	**42.4**	92.0	73.0	81.4	30.9	**56.7**	40.0
test-KBP	Sentence Baseline	11.8	**59.7**	19.9	90.1	65.8	76.0	32.5	19.9	24.7
	Choi et al. (2016) SVM base	27.6	41.2	33.1				36.2	35.5	35.9
	Choi et al. (2016) SVM+ILP	**34.6**	36.8	**35.7**				**45.5**	32.7	**38.1**
	Attentive biLSTM	19.1	30.4	23.5	89.6	80.3	84.7	29.3	41.8	34.4
	Pairwise Attentive biLSTM	21.8	41.6	28.6	91.0	69.6	78.9	23.9	**53.2**	33.0
test-MPQA	Sentence Baseline	10.8	**43.8**	17.4	89.0	74.1	80.9	17.5	3.5	5.9
	Choi et al. (2016) SVM base	**23.2**	16.3	19.2				**28.7**	23.0	25.6
	Choi et al. (2016) SVM+ILP	17.6	24.4	**20.4**				25.2	29.3	**27.1**
	Attentive biLSTM	17.9	23.2	20.2	88.4	87.7	88.1	21.5	17.8	19.4
	Pairwise Attentive biLSTM	18.8	20.5	19.6	89.3	77.6	83.0	15.3	**34.9**	21.3

Table 2: Results on dev-eval, test-KBP, and test-MPQA. For each sentiment class, shows precision, recall, and f1 scores obtained by each model. **Bold** indicates the best score for each label among all five models, whereas underline indicates the best score among only neural models (Sentence, Attentive biLSTM, Pairwise Attentive biLSTM). Note that Choi et al. (2016) did not report results for the 'None' label.

4.1 Sentence Baseline

Socher et al. (2013) introduced a sentence-level RNN model for classifying the sentiment of sentences. The model takes a sentence as input and classifies its sentiment into negative/neutral/positive. We use an adaptation of this model to encapsulate how sentence-level models will perform on our task. In our adaption, we begin by collecting all sentences in which the holder and target entities co-occur. We then classify each of the sentences according to Socher et al. (2013)'s model. We classify the pair's sentiment as 'none' if the entities do not co-occur in any sentence or if all sentences in which the entities co-occur in are classified as 'neutral.' We classify the pair's sentiment as 'positive' if at least one sentence in which the entities co-occur in is classified as 'positive.' Otherwise, we classify the pair's sentiment as 'negative.' Domain differences are the motivation for this split proportion, specifically the fact that positive labels can be up to two times more frequent than negative labels in our dataset.

Note that such a model assumes that any sentence in which holder and target entities co-occur express a single sentiment, and that the sentiment expressed in the sentence is the sentiment between the entity pair. Both assumptions represent implicit limitations in sentence-level models. Pipelined approaches to sentence-level fine-grained sentiment analysis, which first isolate sentiments before isolating the holders and targets of that sentiment (Kim and Hovy, 2006), also make these assumptions.

4.2 SVM and SVM+ILP Baseline

As additional baselines for comparison, we report positive and negative F1 scores from Choi et al. (2016), which is state-of-the-art for document-level entity-entity sentiment extraction. In Choi et al. (2016)'s SVM+ILP model, social science theories were encoded as soft ILP constraints on top of a base SVM pairwise classifier. We report results from both the SVM base classifier and the SVM+ILP classifier with encoded constraints. Our model, which does not use social science constraints, does not ultimately outperform the final SVM+ILP classifier. However, it comes to within 2-3 points of the SVM base classifier. It also has significantly higher recall than either classifier.

4.3 Ablations on PABL

Results from ablating our model are shown in table 3. We report performance on the development data (dev-eval) for our ablation study. In particular, we focus on the effect of removing features. Features for number of mentions (ϕ_f and ϕ_r) contribute the most to the final results.

	NEG	NONE	POS
PABL (Full)	42.4	81.4	40.0
– # mentions rank (ϕ_r)	22.0	77.2	33.4
– # mentions (ϕ_f)	33.0	77.7	38.0
– co-occurrence (ϕ_c)	33.8	79.8	38.0
– sentence (ϕ_s)	37.6	79.1	37.9

Table 3: Ablations. Ablated various features and reported results on dev-eval. Each feature is important in contributing to the final F1 score.

Mention rank features (ϕ_r) contributed nearly 20 points to the f1 score on the negative label, and 4-7 points on none/negative labels. Mention frequency features (ϕ_f) contributed 11 points on the negative label and 2-4 points on none/negative labels. This features are extremely predictive for discerning whether or not entity pairs hold sentiment, which was by far the most challenging task for the model, thus explaining why our model benefit so much from such feature. Co-occurrence features (ϕ_c) are the next largest contributors to the final results. This feature also helps the model determine whether sentiment is present. Analysis on the development data shows that pairs labelled "none" are nearly 2 times less likely to co-occur in any sentence. Contrarily, examples labelled "positive" and "negative" are 2.5-3.4 times *more* likely to co-occur in at least one sentence. Finally, despite being the least helpful, sentence baseline features (ϕ_s) still contributed 2-5 points to the f1 score. More thorough investigations reveals that this feature boosts recall on negative examples, while boosting precision on positive examples (at the sacrifice of recall). This trend is consistent with the performance of the sentence baseline reported in table 2, whereby the sentence baseline achieved the highest recall on negative examples, and the highest precision on positive examples.

4.4 Error Analysis of Neural Models

We use examples from dev-eval for error analysis. We perform error analysis on both the sentence-level baseline and our LSTM-based models to see how our models' errors compares to those made by sentence-level models, and whether it avoids the limitations posed by these models.

Sentence Baseline. The most major source of error in the sentence baseline comes from misalignment between the sentiment of the sentence and the actual sentiment between entities. We look at entity pairs which co-occurred in one sentence, since for these pairs, the sentiment of the entity pair is exactly determined by the sentiment of the sentence that they co-occur in. Of the entity pairs which co-occur once, 87.5% of them co-occurred in negatively-classified sentences, 4.4% of them co-occurred in a neutral-classified sentences, and the rest co-occurred in positively-classified sentences. However, only 7.7% of these examples should have been labelled negatively, while as many as 73.5% of them should have been labelled "none." This misalignment alone accounted for at least 72.8%[2] of all mistakes made by the mode.

The second most common source of error is due to the model's inability to learn sentiment between sentiment pairs which do not co-occur in any sentence. In our data, at least 23.0% of negatively-labelled pairs and 28.0% of positively-labelled pairs do not co-occur in a sentence. Due to its inherent limitation as a sentence-level model, the sentence baseline is unable to classify these pairs correctly. This accounts for 14% of all errors.

LSTM-Based Models: ABL and PABL. As observed, one limitation of the sentence baseline is its inability to classify non-co-occurring pairs as holding sentiment. On the other hand, LSTM-based models do actually learn to classify non-co-occurring as positive or negative. In fact, these models have the opposite problem of being overzealous in doing so, with 39% of its predicted negative pairs and 35% of its predicted positive pairs not co-occurring in any sentence. In actuality, only 23% of negative pairs and 28% of positive pairs do not co-occur in any sentence. This mirrors the larger trend whereby LSTM-based models are overeager in predicting pairs as holding sentiment

Incorrectly classifying "none" examples as either "positive" or "negative" accounted for 62.8% and 81.5% of all errors respectively, for both the ABL and PABL model. This trend is shown in table 4. This is also consistent with the precision/recall scores for each label as reported in table 2, with recall being much higher than precision for positive and negative examples, and precision being higher than recall for "none" examples.

The annotations themselves may contribute to the model's poor performance in identifying "none" examples. "None" examples are by far

[2] Even this number is an underestimate as it only accounts for mistakes made by entities which co-occurred once, over the total number of mistakes.

		Label		
		Neg	None	Pos
ABL Prediction	Neg	**84**	213	66
	None	52	**1946**	145
	Pos	38	295	**193**
PABL Prediction	Neg	**130**	332	94
	None	29	**1487**	82
	Pos	15	635	**228**

Table 4: Model predictions vs. actual classifications of example for both LSTM-based models. **Bold** is used to signify the most frequent prediction for each label. The most significant source of error for both models is the classification of "none" examples as "neg" or "pos." This is especially apparent for PABL.

the most difficult and subjective label to annotate. As an example, different documents containing the phrase "[Country] President [Person]" were annotated differently. In some documents, holder = [Person], target = [Country] was labelled "positive," whereas in others (such as the document in the introduction) holder = [Person], target = [Country] was labelled "none." These inconsistent annotations, especially on "none" examples, can harm the performance of our model.

Surprisingly, the vanilla version of the pairwise attentive bidirectional LSTM without any features does not outperform the attentive bidirectional LSTM. However, with the addition of features, this model receives a significant boost in performance.

5 Related Work

Very little research has been done thus far on document-level extraction of entity-to-entity sentiment relations. Of what has been done, Choi et al. (2016) uses a SVM-based model to predict directed opinions in text, and investigates the effect of encoding social science theories into their model. Our work uses neural models, which require less features and thus less preprocessing.

Related work in fine-grained sentiment analysis usually focuses on one or more aspects of the task, such as identifying sentiment expression (İrsoy and Cardie, 2014), identifying sentiment target (Liu et al., 2015), jointly extracting sentiment polarity and target (Mitchell et al., 2013), or jointly extracting sentiment polarity, holder, and target (Katiyar and Cardie, 2016; Yang and Cardie, 2013). The latter two tasks require combining sentiment analysis with NER. In this work we focus solely on classifying sentiment polarity.

Popular approaches to fine-grained sentiment analysis include CRFs (Yang and Cardie, 2012; Mitchell et al., 2013) and CRF+ILP joint models (Yang and Cardie, 2013). More recently, recurrent neural networks, in particular LSTMs, have proven to be effective for this task (Liu et al., 2015; İrsoy and Cardie, 2014). There is some conflicting literature in regards to how LSTMs perform relative to CRFs. İrsoy and Cardie (2014) finds that multi-layer bi-directional LSTMs outperform CRFs on opinion expression extraction, and Liu et al. (2015) finds that LSTMs used in conjugation with word embeddings outperform CRFs on opinion target extraction without the need for feature engineering. On the other hand, Katiyar and Cardie (2016) finds that bi-directional LSTMs do not outperform conventional CRFs for holder and target extraction, but that adding sentence-level and relation-level dependencies to the final layer improves performance to within 1-3% of CRF models. Almost all models in this field operate on the sentence level, where one sentiment is being expressed between one holder and one target. Our work differs by working at the document level, focused on capturing complex networks of sentiments between many different entity pairs.

Our pairwise attentive biLSTM model borrows architectural components from related fields such as co-reference resolution and document-level relation extraction. We represent mention spans by concatenating encoded start and end tokens with an attentive mechanism. These span representations have been used with relative success in co-reference resolution (Lee et al., 2017), with the attention component shown to be particularly useful. The generation of pairwise scores between all mentions of each entity and the use of the LogSumExp aggregation function over these scores were inspired by techniques used in document-level relation extraction, in particular the model introduced by Verga et al. (2018).

6 Conclusion

We present two LSTM-based neural models for document-level entity-entity sentiment analysis. Both models significantly outperform the sentence-level baseline. We find that the sentence-level model performs poorly on this task, suggesting that current research in intra-sentence sentiment analysis may not generalize to document-level texts. We additionally find that LSTM-based models may not perform as well as state-of-the-art SVM models. Neural models are generally overeager in assigning sentiment between entity pairs, producing overly dense sentiment networks for each document.

In the future, we hope to explore means to strategically prune excess sentiment relations. We would also like to explore encoding social science constraints into our model, either by encoding the constraints in the loss function while training or by designing an architecture to explicitly model the constraints. Finally, joint extraction of sentiment polarities, holders, and/or targets, have proven to outperform pipelined approached to fine-grained sentiment analysis (Yang and Cardie, 2013). These are all directions to explore in future work.

Acknowledgments

I would like to thank Eunsol Choi for her invaluable contributions to this work. I would also like to thank Yejin Choi, Ari Holtzmann, and Nelson Liu for their advice and feedback.

References

Eunsol Choi, Hannah Rashkin, Luke S. Zettlemoyer, and Yejin Choi. 2016. Document-level sentiment inference with social, faction, and discourse context. In *ACL*.

Xing Fang and Justin Zhan. 2015. Sentiment analysis using product review data. *Journal of Big Data*, 2(1):5.

Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. *ACL*.

Soo-Min Kim and Eduard Hovy. 2006. Identifying and analyzing judgment opinions. *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 200–207.

Dilek Küçük and Fazli Can. 2018. Stance detection on tweets: An svm-based approach. *CoRR*, abs/1803.08910.

Guillaume Lample, Miguel Ballesteros, Sandeep K Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Pengfei Liu, Shafiq R. Joty, and Helen M. Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*, pages 1433–1443. The Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *EMNLP*.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *CoRR*, abs/1308.6242.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? *CoRR*, abs/1603.06111.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *CoRR*, cs.CL/0409058.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA. Association for Computational Linguistics.

Sotiris K. Tasoulis, Aristidis G. Vrahatis, Spiros V. Georgakopoulos, and Vassilis P. Plagianakos. 2018. Real time sentiment change detection of twitter data streams. *CoRR*, abs/1804.00482.

Patrick Verga, Emma Strubell, and Andrew McCallum. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. *CoRR*, abs/1802.10569.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP*.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *EMNLP-CoNLL*.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL*.