# Modelling Context in Conversations for Sentiment Analysis

**Rohan Jha**
rjha@cs.brown.edu

**Varun Mathur**
vmathur2@cs.brown.edu

## Abstract

We hypothesize that context is helpful for predicting sentiment in the *EmoContext* task. We build and test three models for the task, each with different representations of the context. The first is a baseline that takes in a representation of only the third sentence; the second is a baseline that takes in representations of the three sentences; and the third takes in a representation of the third sentence, in addition to a representation of the first two sentences obtained from a model trained on the objective of predicting the third sentence from the first two. We observe that the third model is outperformed by both baselines, but the first baseline is outperformed by the second. This suggests that context is helpful for this task but doesn't provide evidence that context is effectively captured by the third model.

## Hypothesis

We hypothesize that context is helpful for predicting sentiment; in particular, that in the *EmoContext* task, the objective of predicting the third message from the first two is useful in modeling context, which in turn helps in predicting the sentiment of the third message.

## 1 Background and Motivation

We're participating in the *EmoContext* task for SemEval-2019, in which the objective is to predict the sentiment of the final text in a conversation with three texts (as *happy*, *sad*, *angry*, or *other*). For example, the following conversation (Gupta et al., 2017) would be predicted as *sad*:

1. **User 1:** I texted you last night!
2. **User 2:** Sorry, I didn't see it until now.
3. **User 3:** Why don't you ever answer me :/

And the following (Gupta et al., 2017) would be predicted as *happy*:

1. **User 1:** I had a game today.
2. **User 2:** Did you win?
3. **User 3:** Yeah!! I'm so happy :)

Sentiment is a critical part of computational semantics, which makes this a relevant task. For a machine listener, an understanding of sentiment is indispensable for capturing the semantics of a human utterance. The two statements "Sure, I guess" and "Yes, I can't wait" have similar denotations but different implications about the enthusiasm of the speaker – and we might want the machine's response to differ between the two.

This example is indicative of the difficulty of this task. As noted above, "Sure, I guess" and "Yes, I can't wait" are equivalent in a model-theoretic sense and have the same Fregeian *reference* (Zalta, 2018). However, they're different in their Fregeian *sense*, which results in a difference in their sentiment. It might follow from this distinction that successful models for this task should capture different conceptions of meaning, including sense and reference. In this paper, we claim that context is also an important part of modelling these meanings.

The motivation for our hypothesis comes in part from the organizers' model for this task (SS-LSTM), which they claim "significantly outperforms traditional Machine Learning baselines as well as other off-the-shelf Deep Learning models" (Gupta et al., 2017). We note that in predicting the sentiment of the third text, SS-LSTM does not take into account the first two messages, which suggests that context might not be useful in this task. However, the authors themselves write that SS-LSTM predicts a label of *Sad* – while the con-

text implies *Happy* – for the following conversation (Gupta et al., 2017):

1. **User 1:** I just qualified for the internship
2. **User 2:** WOOT! That's great news. Congratulations!
3. **User 3:** I started crying.

This indicates that models of context might have a role in predicting this example and others.

Our proposal for modelling the context is in line with the distributional hypothesis. We're extending Firth's notion that words can be "[known]... by the company they keep," (Firth, 1962) to the hypothesis that we can define utterances (in terms of the context they provide) by the utterance that follows. The language modeling objective is something of an instantiation of Firth's hypothesis, and our model is inspired by recent work on this objective (Mikolov et al., 2013). In particular, our objective is similar to that in SkipThought (Kiros et al., 2015), with the difference that instead of jointly predicting the utterances that precede and follow a single utterance, we predict the utterance that follows a pair of utterances. We believe this model is more suited to *EmoContext* because we're not interested in the first two sentences in themselves, merely their role in determining the semantics of the third.

## 2 Experimental Design

We test the hypothesis that the task of predicting the third message from the first two will generate a useful representation of context, with respect to predicting the sentiment of the third message. We outline the primary model, **EncodedContext**, and discuss our two baselines, **NoContext** and **SimpleContext**.

Each model takes in the ELMo embeddings (Peters et al., 2018) for the words in the third text – and possibly some representation of the first two texts (see the next section for details) – and outputs a distribution over sentiment classes.

1. In addition to the embeddings for the third text, **EncodedContext** takes in a single vector that jointly represents the first two texts. This vector is taken from hidden state of an encoder-decoder model that's trained on the objective of predicting the third text from the first two.
2. **NoContext** only takes in the embeddings for the third text.
3. **SimpleContext** takes in the embeddings for the first two texts, in addition to those for the third.

The shared model is trained on the training dataset from the *EmoContext* task. The encoder-decoder model in the **EncodedContext** model is trained on conversations scraped from Reddit, following the approach of Yang, et al. (Yang et al., 2018) (more discussion below of the data and training).

We finally evaluate each of the three models by their performance on the *EmoContext* task. This performance is measured in terms of the prediction accuracy and the F1 score.

## 3 Implementation Details

### 3.1 Tokenization and Normalization

Texting data relies heavily on emoji and ASCII smiley usage, and these might contain important signal for sentiment tasks.

We found that a large proportion of the emojis used were part of the "emoticon" subset (a specific Unicode character range), and there were only a few common non-emoticons, such as hearts. We manually created a new class of character called "emotionicons," consisting of all emoticons and other handpicked emojis which we hypothesized were most important for capturing the useful signal. We then developed a function that would convert every single "emotionicon" into a corresponding ASCII smiley.

We developed a regex expression to detect ASCII smileys, for tokenization purposes, but otherwise left these unprocessed. The hope was that the character-level ELMo embeddings would learn to recognize the sequences of characters appropriately.

Our preprocessing steps were thus:

1. Remove all non-emotionicon emojis
2. Convert all emotionicon emojis into ASCII smileys
3. Use a combination of our regex ASCII detector and the `nltk` tokenizer (Loper and Bird, 2002) to tokenize sentences while maintaining smileys as single tokens.

### 3.2 NoContext

ElMo word embeddings are produced for each word in the third sentence, and combined using a pooling layer. The resulting sentence embedding is then passed through a fully connected softmax classifier. The model is trained on the *EmoContext* training set using negative-likelihood loss with the Adam optimizer and a learning rate of 1e-4.

### 3.3 SimpleContext

All messages will be passed through the same ElMo and pooling layers as above in order to obtain sentence embeddings. The three embeddings are then concatenated, and the result is passed through the same fully connected log-softmax classifier. The model is trained on the *EmoContext* training set using negative-likelihood loss with the Adam optimizer and a learning rate of 1e-4, as above.

### 3.4 EncodedContext

We model context within the framework of encoder-decoder models. In our case, the encoder creates a hidden state representing the contents of messages 1 and 2, and the decoder uses that hidden state to generate the third message.

#### 3.4.1 Pretrained Encoder-Decoder Architecture

- **Embedding.** After tokenization and normalization (discussed below), the words in messages 1 and 2 are initialized with their ELMo representations, with a separator <EOS> token:

  $$w_1^1 \dots w_1^N, \texttt{<EOS>}, w_2^1, \dots w_2^M$$

  We settled on using "|" as the <EOS> token because it's uncommonly used otherwise but is often used as a textual separator, which ELMo might have pretrained knowledge of. The hope is that this is initially no worse than a random embedding, and that ELMo will learn to recognize it later on.

- **Encoder.** The encoder is a gated recurrence unit (GRU). A GRU was chosen as it is conceptually simpler and faster to train than an LSTM, while still providing state-of-the-art results in many cases. Given some input embedding $w_i$ and the previous hidden state $h_{i-1}$, the GRU learns the non-linear function:

  $$h_i = f(w_i, h_{i-1})$$

  At the end of the encoding process, we take the final state. This intuitively represents all the information contained in the first two sentences.

  $$h_{enc}$$

- **Decoder.** The decoder is a GRU conditioned on the hidden state of the encoder that attempts to predict the ELMo embeddings of the target word. The initial hidden state of the GRU is set to the final hidden state of the encoder, $h_{enc}$.

  To predict the $i$th word, the decoder takes as input its previous hidden state, $s^{i-1}$ and the previous predicted word $w^{i-1}$. It first computes some representation of the previous word:

  $$w^{i-1} = \alpha w^{i-1}$$

  It then passes this representation along with the hidden state through the GRU, which learns the function:

  $$w^i, s^i = g(w^{i-1}, s^{i-1})$$

  Finally, we learn another function that takes the word $w^i$ into ELMo embedding space:

  $$w_i = \beta w_i$$

- **Objective.** Most implementations of encoder-decoder networks in NMT (Neural Machine Translation) use the final word prediction of $w_i = \texttt{softmax}(\phi x_i)$ to project to a probability distribution in vocab space, where the translation would simply be the one-hot max of this vector.

  Since we are not translating and thus not constrained with this semantic interpretation of $w_i$, we learn $\phi$ to project into ELMo embedding space. We calculate loss for each prediction based on the cosine-distance of the target ELMo embedding $w_3^i$ and the predicted ELMo embedding $w^i$:

  $$1 - \frac{w^i \cdot w_3^i}{\|w^i\|_2 \|w_3^i\|_2}$$

  This novel approach is important for two reasons:

  1. In the text message domain, vocabulary is very ill-defined and thus character-level embeddings such as ELMo are essential

2. This allows our model to generalize: predicting "red" instead of "maroon" will result in small error which intuitively makes sense. This behavior cannot be accomplished using the one-hot formulation.

- **Training.** The Adam optimizer is used, and the learning rate is 1e-4.

  The decoder is trained using "teacher forcing" to cause the model to converge faster. At each timestep we condition on the real target's previous word instead of using the decoder's guess.

- **Data.** We trained on both the Cornell Movie-Dialogs Corpus (Danescu-Niculescu-Mizil and Lee, 2011) and data scraped from Reddit, though the final model is trained only on Reddit data.

  To scrape the Reddit data, we adapt the approach of Yang, et al. (Yang et al., 2018) and Al-Rfou, et al. (Al-Rfou et al., 2016). We search for triples of three comments, in which the first comment is a response to the second comment, and the second comment is a response to the third comment. We then remove triples with more than twenty-five tokens, or with comments that have been deleted or removed.

#### 3.4.2 EncodedContext Architecture

Once we have pretrained the encoder-decoder model, we claim that the final hidden state of the encoder encodes the relevant context of the first two sentences. We concatenate this context vector with the same ELMo and pooling sentence embedding in **NoContext** and then pass the result through a fully connected softmax classifier. The model is trained on the *EmoContext* training set using negative-likelihood loss with the Adam optimizer and a learning rate of 1e-4, as with the baseline models.

## 4 Results

| model | accuracy | F1 |
|---|---|---|
| EncodedContext | 0.636 | 0.512 |
| NoContext | 0.820 | 0.775 |
| SimpleContext | 0.854 | 0.821 |

Table 1: Results

In Table 1, we have the accuracy of each of our three models on the *EmoContext* task, in addition to F1 scores for each of the classes. We follow the organizers' example (Gupta et al., 2017) and compute a simple average of the F1 scores for the classes *Happy*, *Sad*, and *Angry*. With both metrics we find that **SimpleContext** outperforms **NoContext**, but **EncodedContext** is outperformed by the two baselines.

These results aren't directly comparable to those of the organizers' model because we don't have access to their evaluation dataset. However, because our validation data is likely sampled from the same distribution as theirs, we note that our two baselines have a higher F1 score than that of their SS-LSTM model (71.34), which doesn't use the first two texts. The organizers don't report the accuracy of their model.

## 5 Technical Challenges

Our **EncodedContext** model didn't perform as expected.

At first, we observed the model overfitting the training data: test loss was increasing each epoch while training loss was decreasing. We tried manipulating the dropout probabiltiy to no avail. We then 'detached' the encoder from the model such that the weights in the encoder weren't updated in the training of **EncodedContext**. This was to reduce the number of parameters being updated and thereby limit the expressivity of the model and also bring the model's number of parameters to parity with the other baselines. This seemed to solve the overfitting, but the accuracy only increased by a couple percentage points.

We then experimented with learning rates, numbers of hidden layers, and sizes of hidden layers, in the hopes that the model just needed more classification power, capability to escape local minima, etc. We also tried increasing the size of the training data for the Encoder-Decoder; we switched to training on 200k examples from Reddit, as opposed to 50k examples from the Cornell Movie-Dialogs Corpus. Neither of these strategies were successful.

Then as a sanity check, we substituted the encodings in **EncodedContext** with random tensors and found that the model performed much worse. This suggested that the encodings added *some* signal, but much less signal then noise, as evidenced by the relative performance of **EncodedContext**

and **NoContext**. It also indicated that the model doesn't learn to ignore the encodings.

The next step would be to iterate on the Encoder-Decoder to improve the context encodings. Some possibilities are to add an attention mechanism over the hidden states in the decoder, or to revert to a more traditional language-modelling objective, in which the decoding is a sentence, as opposed to a vector.

## 6 Conclusion

Although **EncodedContext** didn't perform well, we note that **NoContext** was outperformed by **SimpleContext**. Therefore, while we find no evidence that **EncodedContext** was an effective model of context, we have evidence for the claim that context is helpful for predicting sentiment in the *EmoContext* task.

## References


Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2016. Conversational contextual cues: The case of personalization and history for response ranking. *CoRR*, abs/1606.00372.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

John Firth. 1962. *Studies in Linguistic Analysis*.

Umang Gupta et al. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *arXiv preprint arXiv:1707.06996*.

Ryan Kiros et al. 2015. Skip-thought vectors. *NIPS*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov et al. 2013. Distributed representations of words and phrases and their compositionality. *NIPS*.

Matthew Peters et al. 2018. Deep contextualized word representations. *NAACL*.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. *CoRR*, abs/1804.07754.

Edward N. Zalta. 2018. Gottlob frege. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, summer 2018 edition. Metaphysics Research Lab, Stanford University.