Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks

Ji Young Lee∗
MIT
jjylee@mit.edu

Franck Dernoncourt∗
MIT
francky@mit.edu

Published as a conference paper at NAACL 2016

Abstract
Recent approaches based on artificial neural networks (ANNs) have shown promising results for short-text classification. However, many short texts occur in sequences (e.g., sentences in a document or utterances in a dialog), and most existing ANN-based systems do not leverage the preceding short texts when classifying a subsequent one. In this work, we present a model based on recurrent neural networks and convolutional neural networks that incorporates the preceding short texts. Our model achieves state-of-the-art results on three different datasets for dialog act prediction.

1 Introduction
Short-text classification is an important task in many areas of natural language processing, including sentiment analysis, question answering, or dialog management. Many different approaches have been developed for short-text classification, such as using Support Vector Machines (SVMs) with rule-based features (Silva et al., 2011), combining SVMs with naive Bayes (Wang and Manning, 2012), and building dependency trees with Conditional Random Fields (Nakagawa et al., 2010). Several recent studies using ANNs have shown promising results, including convolutional neural networks (CNNs) (Kim, 2014; Blunsom et al., 2014; Kalchbrenner et al., 2014) and recursive neural networks (Socher et al., 2012).

Most ANN systems classify short texts in isolation, i.e., without considering preceding short texts.

However, short texts usually appear in sequence (e.g., sentences in a document or utterances in a dialog), therefore using information from preceding short texts may improve the classification accuracy. Previous works on sequential short-text classification are mostly based on non-ANN approaches, such as Hidden Markov Models (HMMs) (Reithinger and Klesen, 1997), (Stolcke et al., 2000), maximum entropy (Ang et al., 2005), and naive Bayes (Lendvai and Geertzen, 2007).

Inspired by the performance of ANN-based systems for non-sequential short-text classification, we introduce a model based on recurrent neural networks (RNNs) and CNNs for sequential short-text classification, and evaluate it on the dialog act classification task. A dialog act characterizes an utterance in a dialog based on a combination of pragmatic, semantic, and syntactic criteria. Its accurate detection is useful for a range of applications, from speech recognition to automatic summarization (Stolcke et al., 2000). Our model achieves state-of-the-art results on three different datasets.

2 Model
Our model comprises two parts. The first part generates a vector representation for each short text using either the RNN or CNN architecture, as discussed in Section 2.1 and Figure 1. The second part classifies the current short text based on the vector representations of the current as well as a few preceding short texts, as presented in Section 2.2 and Figure 2.

We denote scalars with italic lowercases (e.g., $k, b_f$), vectors with bold lowercases (e.g., $s, x_i$), and matrices with italic uppercases (e.g., $W_f$). We

∗ These authors contributed equally to this work.
use the colon notation \( v_{i:j} \) to denote the sequence of vectors \( (v_i, v_{i+1}, \ldots, v_j) \).

## 2.1 Short-text representation

A given short text of length \( \ell \) is represented as the sequence of \( m \)-dimensional word vectors \( x_{1:\ell} \), which is produced by the RNN or CNN model to produce the \( n \)-dimensional short-text representation \( s \).

### 2.1.1 RNN-based short-text representation

We use a variant of RNN called Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). For the \( t^{th} \) word in the short-text, an LSTM takes as input \( x_t, h_{t-1}, c_{t-1} \) and produces \( h_t, c_t \) based on the following formulas:

\[
\begin{align*}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
c_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_{t+1} &= f_t \odot c_{t-1} + i_t \odot c_t \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
h_t &= o_t \odot \tanh(c_t)
\end{align*}
\]

where \( W_j \in \mathbb{R}^{h \times m} \), \( U_j \in \mathbb{R}^{n \times n} \) are weight matrices and \( b_j \in \mathbb{R}^n \) are bias vectors, for \( j \in \{i, f, c, o\} \).

The symbols \( \sigma(\cdot) \) and \( \tanh(\cdot) \) refer to the element-wise sigmoid and hyperbolic tangent functions, and \( \odot \) is the element-wise multiplication. \( h_0 = c_0 = 0 \).

In the pooling layer, the sequence of vectors \( h_{1:\ell} \) output from the RNN layer are combined into a single vector \( s \in \mathbb{R}^n \) that represents the short-text, using one of the following mechanisms: last, mean, and max pooling. Last pooling takes the last vector, i.e., \( s = h_{\ell} \), mean pooling averages all vectors, i.e., \( s = \frac{1}{\ell} \sum_{t=1}^{\ell} h_t \), and max pooling takes the element-wise maximum of \( h_{1:\ell} \).

### 2.1.2 CNN-based short-text representation

Using a filter \( W_f \in \mathbb{R}^{h \times m} \) of height \( h \), a convolution operation on \( h \) consecutive word vectors starting from \( t^{th} \) word outputs the scalar feature

\[
c_t = \text{ReLU}(W_f \bullet X_{t:t+h-1} + b_f)
\]

where \( X_{t:t+h-1} \in \mathbb{R}^{h \times m} \) is the matrix whose \( i^{th} \) row is \( x_i \in \mathbb{R}^m \), and \( b_f \in \mathbb{R} \) is a bias. The symbol \( \bullet \) refers to the dot product and \( \text{ReLU}(\cdot) \) is the element-wise rectified linear unit function.

We perform convolution operations with \( n \) different filters, and denote the resulting features as \( c_t \in \mathbb{R}^n \), each of whose dimensions comes from a distinct filter. Repeating the convolution operations
for each window of $h$ consecutive words in the short-text, we obtain $c_{1:t-h+1}$. The short-text representation $s \in \mathbb{R}^n$ is computed in the max pooling layer, as the element-wise maximum of $c_{1:t-h+1}$.

2.2 Sequential short-text classification

Let $s_i$ be the $n$-dimensional short-text representation given by the RNN or CNN architecture for the $i^{th}$ short text in the sequence. The sequence $s_{i-d_1:d_2}:i$ is fed into a two-layer feedforward ANN that predicts the class for the $i^{th}$ short text. The hyperparameters $d_1, d_2$ are the history sizes used in the first and second layers, respectively.

The first layer takes as input $s_{i-d_1:d_2}:i$ and outputs the sequence $y_{i-d_2}:i$, defined as

$$y_j = \tanh \left( \sum_{d=0}^{d_2} W_{-d}s_{j-d} + b_1 \right), \forall j \in [i-d_2,i]$$

where $W_0, W_{-1}, W_{-2} \in \mathbb{R}^{k \times n}$ are the weight matrices, $b_1 \in \mathbb{R}^k$ is the bias vector, $y_j \in \mathbb{R}^k$ is the class representation, and $k$ is the number of classes for the classification task.

Similarly, the second layer takes as input the sequence of class representations $y_{i-d_2}:i$ and outputs $z_i \in \mathbb{R}^k$:

$$z_i = \text{softmax} \left( \sum_{j=0}^{d_2} W_{-j}y_{i-j} + b_2 \right)$$

where $U_0, U_{-1}, U_{-2} \in \mathbb{R}^{k \times k}$ and $b_2 \in \mathbb{R}^k$ are the weight matrices and bias vector.

The final output $z_i$ represents the probability distribution over the set of $k$ classes for the $i^{th}$ short-text: the $j^{th}$ element of $z_i$ corresponds to the probability that the $i^{th}$ short-text belongs to the $j^{th}$ class.

3 Datasets and Experimental Setup

3.1 Datasets

We evaluate our model on the dialog act classification task using the following datasets:

- **DSTC 4**: Dialog State Tracking Challenge 4 (Kim et al., 2015; Kim et al., 2016).
- **MRDA**: ICSI Meeting Recorder Dialog Act Corpus (Janin et al., 2003; Shriberg et al., 2004). The 5 classes are introduced in (Ang et al., 2005).
- **SwDA**: Switchboard Dialog Act Corpus (Jurafsky et al., 1997).

For MRDA, we use the train/validation/test splits provided with the datasets. For DSTC 4 and SwDA, only the train/test splits are provided.\(^1\) Table 1 presents statistics on the datasets.

| Dataset   | $|C|$ | $|V|$ | Train | Validation | Test |
|-----------|-----|-----|-------|------------|------|
| DSTC 4    | 89  | 6k  | 24 (21k) | 5 (5k) | 6 (6k) |
| MRDA      | 5   | 12k | 51 (78k) | 11 (16k) | 11 (15k) |
| SwDA      | 43  | 20k | 1003 (193k) | 112 (23k) | 19 (5k) |

\(^1\)All train/validation/test splits can be found at [https://github.com/Franck-Dernoncourt/naacl2016](https://github.com/Franck-Dernoncourt/naacl2016)

3.2 Training

The model is trained to minimize the negative log-likelihood of predicting the correct dialog acts of the utterances in the train set, using stochastic gradient descent with the Adadelta update rule (Zeiler, 2012). At each gradient descent step, weight matrices, bias vectors, and word vectors are updated. For regularization, dropout is applied after the pooling layer, and early stopping is used on the validation set with a patience of 10 epochs.

4 Results and Discussion

To find effective hyperparameters, we varied one hyperparameter at a time while keeping the other ones fixed. Table 2 presents our hyperparameter choices.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Choice</th>
<th>Experiment Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM output dim. $(n)$</td>
<td>100</td>
<td>50 – 1000</td>
</tr>
<tr>
<td>LSTM pooling</td>
<td>max</td>
<td>max, mean, last</td>
</tr>
<tr>
<td>LSTM direction</td>
<td>unidir.</td>
<td>unidir., bidir.</td>
</tr>
<tr>
<td>CNN num. of filters $(n)$</td>
<td>500</td>
<td>50 – 1000</td>
</tr>
<tr>
<td>CNN filter height $(h)$</td>
<td>3</td>
<td>1 – 10</td>
</tr>
<tr>
<td>Dropout rate $(h)$</td>
<td>0.5</td>
<td>0 – 1</td>
</tr>
<tr>
<td>Word vector dim. $(m)$</td>
<td>200, 300</td>
<td>25 – 300</td>
</tr>
</tbody>
</table>

Table 2: Experiments ranges and choices of hyperparameters. Unidir refers to the regular RNNs presented in Section 2.1.1, and bidir refers to bidirectional RNNs introduced in (Schuster and Paliwal, 1997).
word vectors pretrained with GloVe on Twitter (Pennington et al., 2014) for MRDA and SwDA, as these choices yielded the best results among all publicly available word2vec, GloVe, SENNA (Collobert, 2011; Collobert et al., 2011) and RNNLM (Mikolov et al., 2011) word vectors.

The effects of the history sizes $d_1$ and $d_2$ for the short-text and the class representations, respectively, are presented in Table 3 for both the LSTM and CNN models. In both models, increasing $d_1$ while keeping $d_2 = 0$ improved the performances by 1.3-4.2 percentage points. Conversely, increasing $d_2$ while keeping $d_1 = 0$ yielded better results, but the performance increase was less pronounced: incorporating sequential information at the short-text representation level was more effective than at the class representation level.

Using sequential information at both the short-text representation level and the class representation level does not help in most cases and may even lower the performances. We hypothesize that short-text representations contain richer and more general information than class representations due to their larger dimension. Class representations may not convey any additional information over short-text representations, and are more likely to propagate errors from previous misclassifications.

Table 4 compares our results with the state-of-the-art. Overall, our model shows competitive results, while requiring no human-engineered features. Rigorous comparisons are challenging to draw, as many important details such as text preprocessing and train/valid/test split may vary, and many studies fail to perform several runs despite the randomness in some parts of the training process, such as weight initialization.

5 Conclusion

In this article we have presented an ANN-based approach to sequential short-text classification. We demonstrate that adding sequential information improves the quality of the predictions, and the performance depends on what sequential information is used in the model. Our model achieves state-of-the-art results on three different datasets for dialog act prediction.

<table>
<thead>
<tr>
<th>$d_1$</th>
<th>$d_2$</th>
<th>LSTM</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>63.1</td>
<td>64.1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>65.8</td>
<td>64.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>65.7</td>
<td>65.4</td>
</tr>
</tbody>
</table>

Table 3: Accuracy (%) on different architectures and history sizes $d_1$, $d_2$. For each setting, we report average (minimum, maximum) computed on 5 runs. Sequential classification ($d_1 + d_2 > 0$) outperforms non-sequential classification ($d_1 = d_2 = 0$). Overall, the CNN model outperformed the LSTM model for all datasets, albeit by a small margin except for SwDA. We also tried a variant of the LSTM model, gated recurrent units (Cho et al., 2014), but the results were generally lower than LSTM.

<table>
<thead>
<tr>
<th>Model</th>
<th>DSTC4</th>
<th>MRDA</th>
<th>SwDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>65.5</td>
<td>84.6</td>
<td>73.1</td>
</tr>
<tr>
<td>LSTM</td>
<td>66.2</td>
<td>84.3</td>
<td>69.6</td>
</tr>
<tr>
<td>Majority class</td>
<td>25.8</td>
<td>59.1</td>
<td>33.7</td>
</tr>
<tr>
<td>SVM</td>
<td>57.0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Graphical model</td>
<td>–</td>
<td>81.3</td>
<td>–</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>–</td>
<td>82.0</td>
<td>–</td>
</tr>
<tr>
<td>HMM</td>
<td>–</td>
<td>71.0</td>
<td>–</td>
</tr>
<tr>
<td>Memory-based Learning</td>
<td>–</td>
<td>72.3</td>
<td>–</td>
</tr>
<tr>
<td>Interlabeler agreement</td>
<td>–</td>
<td>84.0</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 4: Accuracy (%) of our models and other methods from the literature. The majority class model predicts the most frequent class. SVM: (Dernoncourt et al., 2016). Graphical model: (Ji and Bilmes, 2006). Naive Bayes: (Lendvai and Geertzen, 2007). HMM: (Stolcke et al., 2000). Memory-based Learning: (Rotaru, 2002). All five models use features derived from transcribed words, as well as previous predicted dialog acts except for Naive Bayes. The interlabeler agreement could be obtained only for SwDA. For the CNN and LSTM models, the presented results are the test set accuracy of the run with the highest accuracy on the validation set.
References


