Sequential Short Text Classification with Recurrent Neural Networks

Ji Young Lee
MIT
77 Massachusetts Ave
Cambridge, MA 02139, USA
jjylee@mit.edu

Abstract

Short text classification is an old problem in natural language processing, and has various applications from question answering systems to sentiment classification. Recent approaches based on artificial neural networks have shown competitive results compared to traditional approaches such as support vector machine (SVM) and maximum entropy. However, many existing systems for short text classification do not incorporate the context of the short texts. In this work, we present a system based on recurrent neural networks, which sequentially classifies short texts based on the context.

We evaluate our approach on the speech act predicition task using the Dialog State Tracking Challenge 4 data set, which consists of 35 dialogs annotated with speech acts for each utterance. We show that incorporating contextual information improves the performance of classification over non-contextual models, i.e., models that classify each sentence separately without using the context. In addition, we show that our approach yields higher performances than models based on logistic regression and SVM that also incorporate the context.[1]

1 Introduction

Text classification is an important task in many areas of natural language processing, including sentiment analysis, question answering, and dialog management. For example, the goal of text classification may be categorizing texts based on sentiments, classifying questions according to semantic categories, or classifying utterances in dialogues according to speech acts.

There have been many different approaches to tackle the problem of text classification, such as using Support Vector Machine (SVM) with rule-based features (Silva et al., 2011), a combination of SVM with naive Bayes (Wang and Manning, 2012), and dependency tree with Conditional Random Fields (Nakagawa et al., 2010). Following the developments in the field of artificial neural networks (ANNs), more recent approaches include convolutional neural networks (CNNs) (Kim, 2014), (Blunsom et al., 2014), and recursive neural networks (RNNs) (Socher et al., 2012). These ANN-based approaches not only eradicate the need to handcraft features, but also circumvent the data sparsity issues by replacing n-grams by dense word embeddings, and often outperform traditional approaches.

However, most text classification systems focus on classifying texts that arise without any contexts. This may be due to the fact that many existing data sets inherently do not need any context for the classification task at hand. For example, in the question corpus from Text REtrieval Conference (TREC), the main goal is to classify each individual question according to semantic categories; the questions are provided without any context, since they already contain all the information within themselves necessary for the classification.

Nevertheless, in many other text classification tasks, the contexts in which the texts appear may be informative or even be required for classify-

[1] https://github.mit.edu/jjylee/textclassifier

ing the text. Examples of such systems include sentiment, topic, or rhetorical status classification of sentences in a document, and dialog act classification of utterances in a dialog. Compared to its non-contextual counterpart abounding with ANN-based approaches, previous works for contextual text classification are mostly based on traditional approaches, such as Hidden Markov Models (HMMs) (Reithinger and Klesen, 1997), (Stolcke et al., 2000), decision trees (Verbree et al., 2006), maximum entropy (Ang et al., 2005), and naive Bayes (Lendvai and Geertzen, 2007).

Inspired by the performance of ANN-based systems for non-contextual text classification, we propose a model based on recurrent neural networks (RNNs) for sequential short text classification. In particular, our model employs an RNN-based architecture on top of word vectors to generate sentence representations, and classifies sentences based on the representations of both the current and previous sentences. We evaluate our system on speech act classification task for dialog state tracking challenge 4 (DSTC4) dataset. We show that our system outperforms the baselines that use logistic regression (LR) and SVM, and analyze the impact of using sequential versus non-contextual model, as well as of changing various hyperparameters.

2 Model

The model architecture for non-contextual text classification is based on RNN, as shown in Figure 1. The system first generates the vector representation of a sentence via RNN that takes as input the embeddings of the words in the sentence. The vector representation is then fed into a feed forward neural network with softmax activation function, whose output units correspond to the probability of belonging to each class. This model is elaborated in Section 2.1.

Our sequential text classification system shown in Figure 2 is an extension of the non-contextual counterpart, where the vector representation of each sentence is generated from the same RNN-based architecture. The difference is that instead of using only the vector representation of the sentence in consideration, it incorporates the context by utilizing the vector representations of both the target sentence and the previous sentence. The sequential model is discussed in Section 2.2.

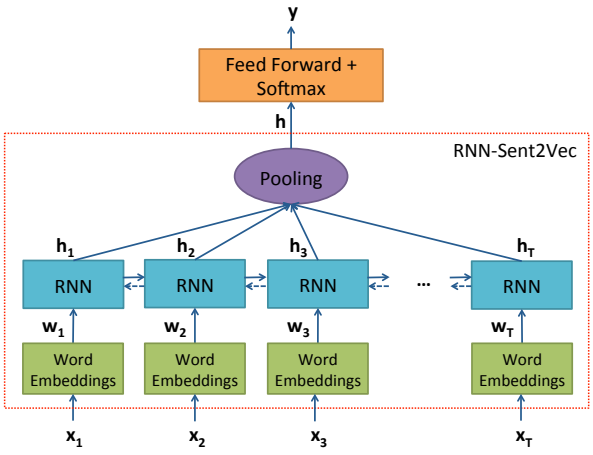

Figure 1: Non-contextual model using RNNs

2.1 Sentence representation using RNN

Let $\mathbf{x}_t \in \mathbb{R}^d$ be the one-hot vector of t-th word in a sentence of length T. In the word embedding layer, the corresponding word vector $\mathbf{w}_t \in \mathbb{R}^m$ can be obtained as $\mathbf{w}_t = W_v \mathbf{x}_t$, where $W_v \in \mathbb{R}^{m \times d}$ is the word vector matrix containing word vectors at each row. Word vector matrix W can be initialized randomly or with the pretrained word vectors.

Given the sequence of word vectors $(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T)$, the sequence of hidden state vectors $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T)$, is computed by the RNN layer. Specifically, the variants of RNN called the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or the Gated Recurrent Unit (GRU) (Cho et al., 2014) may be used. LSTM is defined as follows:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(W_i \mathbf{w}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(W_f \mathbf{w}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\tilde{\mathbf{c}}_t &= \tanh(W_c \mathbf{w}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(W_o \mathbf{w}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
$$

where $W_i, W_f, W_c, W_o \in \mathbb{R}^{n \times m}$, $U_i, U_f, U_c, U_o \in \mathbb{R}^{n \times n}$ are weight matrices and $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^n$ are bias vectors used in the input gate, forget gate, cell state, and output gate calculations, respectively. $\odot$ denotes the element-wise multiplication.

GRU is a simplified version of LSTM, without compromising the performances:

$$\begin{aligned}
\mathbf{r}_t &= \sigma(W_r \mathbf{w}_t + U_r \mathbf{h}_{t-1} + \mathbf{b}_r) \\
\tilde{\mathbf{h}}_t &= \tanh(W_h \mathbf{w}_t + U_h(\mathbf{r}_t \odot \mathbf{h}_{t-1} + \mathbf{b}_h) \\
\mathbf{z}_t &= \sigma(W_z \mathbf{w}_t + U_z \mathbf{h}_{t-1} + \mathbf{b}_z) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t
\end{aligned}$$

where $W_r, W_h, W_z \in \mathbb{R}^{n \times m}$, $U_r, U_h, U_z \in \mathbb{R}^{n \times n}$ are weight matrices and $\mathbf{b}_r, \mathbf{b}_h, \mathbf{b}_z \in \mathbb{R}^n$ are bias vectors used in the input gate, forget gate, cell state, and output gate calculations, respectively. $\odot$ denotes the element-wise product of vectors.

The RNN could be monodirectional or bidirectional. Monodirectional (forward) RNN generates the hidden states by feeding the input sequence in order from $\mathbf{w_1}$ to $\mathbf{w_T}$. On the other hand, a bidirectional RNN consists of a forward RNN and a backward RNN, where the forward RNN calculates the forward hidden states $(\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_T)$ in the same way as the monodirectional RNN, and the backward RNN calculates the backward hidden states $(\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_T)$ by feeding the input sequence in the backward order, from $\mathbf{w_T}$ to $\mathbf{w_1}$. The final hidden state is obtained by concatenating the forward and the backward hidden state, i.e., $\mathbf{h}_t = (\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t)$. Note that if bidirectional RNN is used, the dimension of the resulting hidden states will be twice compared to monodirectional RNN.

The sequence of hidden vectors $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T)$ output from the RNN layer are combined into a single vector $\mathbf{h} \in \mathbb{R}^n$ in the pooling layer. The pooling layer has four different mechanisms: last pooling, mean pooling, max pooling, and content-based pooling. Last pooling simply takes the last output, i.e., $\mathbf{h} = \mathbf{h}_T$, whearas mean pooling averages all outputs, i.e., $\mathbf{h} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{h}_t$, and max pooling takes the element-wise maximum of the outputs.

Content-based pooling is inspired by the attention-based mechanism.[2] In this case, the vectors are combined by taking the convex combination $h = \sum_{t=1}^{T} \alpha_t h_t$, where the coefficients $\alpha_t, t = 1, \ldots, T$ are determined by

$$\begin{aligned}
e_t &= \mathbf{v}_p^T \tanh(W_p \mathbf{h}_t + \mathbf{b}_p), \\
\alpha_t &= \frac{\exp(e_t)}{\sum_{t=1}^{T} \exp(e_t)},
\end{aligned}$$

where $\mathbf{v}_p \in \mathbb{R}^l$ and $W_p \in \mathbb{R}^{l \times n}$ are the weights and $\mathbf{b}_p \in \mathbb{R}^l$ is the bias vector. In words, $(\alpha_1, \ldots, \alpha_T)$ can be obtained by first computing the scalar output e_t of feed forward neural network with 1 hidden layer from each h_t, and then taking the softmax of $(e_1, \ldots, e_T)$.

The single vector $\mathbf{h}$ output from the pooling layer can be interpreted as the vector representation of the input sentence. The architecture discussed above including word2vec, RNN, and pooling layer are called RNN-Sent2Vec (RNN-S2V), since it converts each sentence $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ into the vector representation $\mathbf{h}$.

In order to classify the sentence using the vector representation $\mathbf{h}$ output from RNN-S2V, a feedforward neural network with softmax activation function takes $\mathbf{h}$ as input and generates the output $\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{R}^k$, where k is the number of classes for the classification task:

$$\mathbf{y} = \text{softmax}(W_{ff}\mathbf{h} + \mathbf{b}_{ff}) \tag{1}$$

where $W_{ff} \in \mathbb{R}^{k \times n}$ and $\mathbf{b}_{ff} \in \mathbb{R}^k$ are the weight and bias vectors. Then y_i represents the probability that the sentence belongs to i-th class.

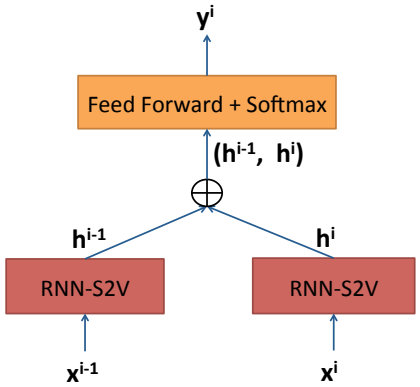

Figure 2: Sequential text classification system

[2] The attention-based mechanisms are commonly used in sequence-to-sequence prediction tasks, such as translation or speech recognition. Such mechanisms are useful to predict which part of the input sequence should be considered to generate each part of the output sequence. Since output is not a sequence in our classification setting, we only use the content of the input sequence (instead of both the input and the output) in order to calculate the coefficients for the convex combination.

	DSTC 4	
	Category	Attribute
T: Can you recommend a hotel?	QUESTION	RECOMMEND
G: Alright.	RESPONSE	POSITIVE
G: On Orchard Road there are a number of hotels.	FOLLOW	INFO
T: Okay.	FOLLOW	ACKNOWLEDGE

Table 1: Sample utterances annotated with dialog acts from the DSTC 4 data set

2.2 Context-based sequential classification

For the sequential classification system, the context of each sentence is considered by using both the target sentence and the previous sentence, as the name 'bigram' context signifies.

Let $\mathbf{x}^i = (\mathbf{x}_1^i, \ldots, \mathbf{x}_{T_i}^i)$ and $\mathbf{x}^{i-1} = (\mathbf{x}_1^{i-1}, \ldots, \mathbf{x}_{T_{i-1}}^{i-1})$ be the sequences of one-hot vector of words in the target sentence and the previous sentence, respectively. RNN-S2V outputs the vector representation $\mathbf{h}^i$ and $\mathbf{h}^{i-1}$ corresponding to the target sentence $\mathbf{x}^i$ and the previous sentence $\mathbf{x^{i-1}}$, respectively.

The sentence representations are concatenated, i.e., $\mathbf{h} = (\mathbf{h}^{i-1}, \mathbf{h}^i)$, and fed into a feedforward network with softmax activation function, which generates the output $\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{R}^k$, where k is the number of classes for the classification task (See equation (1)[3]). The output y_i represents the probability that the sentence belongs to i-th class.

3 Data sets and Experimental Setup

3.1 Data set

We assess our model on the Dialog State Tracking Challenge 4 (DSTC 4) data set, which consists of 35 transcribed Skype dialogs (31,034 utterances and 273,580 words) between a tourist who plans to visit Singapore and a guide. Each utterance is labeled with one or several dialog acts. A dialog act classifies an utterance based on a combination of pragmatic, semantic, and syntactic criteria. Its accurate detection is useful to a range of applications, from speech recognition to automatic summarization (Stolcke et al., 2000).

In the DSTC 4 data set, each dialog act consists of one category (out of 4) and one attribute (out of 22). There are $4 \times 22 = 88$ different dialog acts. The categories are QUESTION, RESPONSE, INITIATIVE, FOLLOW and the attributes are ACKNOWLEDGE, CANCEL, CLOSING, COMMIT, CONFIRM, ENOUGH, EXPLAIN, HOW-MUCH, HOW-TO, INFO, NEGATIVE, OPENING, POSITIVE, PREFERENCE, RECOMMEND, THANK, WHAT, WHEN, WHERE, WHICH, WHO. Table 1 shows sample utterances and the corresponding dialog acts. The data set is described in more details in (Kim et al., 2016; Kim et al., 2015a).

The dialog act classification was part of a pilot task in DSTC 4, for which the official train/test set split was provided. In our experiments, we follow the official split, where the test set comprises 6 dialogs (5615 utterances). For the validation set, we used the second half of the DSTC 4 main task's test set (5 dialogs, 4539 utterances)[4].

3.2 Baselines

We compare our model against two baseline classifiers that uses LR and SVM, which were the best performing systems submitted to the DSTC 4 (Dernoncourt et al., 2016) pilot task.[5] The features of the baseline classifiers are the 5000 most common unigrams, bigrams, trigrams. The classifiers can be trained as a non-sequential model where the features are based only on the current utterance, or a sequential model where features are computed for both the target and the previous utterance in order to account for the context, in an analogous manner as the non-sequential and the sequential RNN-based models discussed in Section 2.1 and 2.2, respectively.

[3]The only difference with the noncontextual classification system here is the dimension of the weights: $W_{ff} \in \mathbb{R}^{k \times \mathbf{2}n}$.

[4]The main task of DSTC 4 had a different official train/test data split (20/9 dialogs). These 29 dialogs were used as the training data of the pilot task.

[5]The baselines systems were slightly modified to match the experimental setup of our model.

	Non-sequential			Sequential		
	Min	Average	Max	Min	Average	Max
Baseline LR	–	0.5340	–	–	0.5441	–
Baseline SVM	–	0.5506	–	–	0.5573	–
LSTM	0.6331	0.6376	0.6416	0.6436	0.6529	0.6587
GRU	X	X	X	0.6442	0.6484	0.6514

Table 2: Results for the two baselines (LR and SVM) and the RNN models (LSTM and GRU). For the baseline models, non-sequential case uses features on only the current utterance, while sequential case uses features on both the current and the previous utterances. For RNN-based models, non-sequential refers to the model discussed in Section 2.1, sequential refers to the model in Section 2.2. X corresponds to the values that could not be obtained due to time constraints.

	Monodirectional			Bidirectional		
	Min	Average	Max	Min	Average	Max
Last	0.6436	0.6529	0.6587	0.6450	0.6505	0.6581
Mean	0.6416	0.6460	0.6508	0.6423	0.6454	0.6495
Max	0.6593	0.6621	0.6663	0.6579	0.6606	0.6634
Content-based	0.6450	0.6484	0.6516	0.6511	0.6519	0.6526

Table 3: Experiments on monodirectional vs bidirectional LSTMs and pooling methods (last, mean, max, and content-based)

	Min	Average	Max
LSTM output dimension 100	0.6555	0.6596	0.6661
LSTM output dimension 250	0.6506	0.6545	0.6585
LSTM output dimension 500	0.6436	0.6529	0.6587
Randomly initialized word embeddings	0.6456	0.6491	0.6518
Dropout regularization	0.6519	0.6575	0.6626

Table 4: Experiments on output dimensions of LSTM, random initialization of word embeddings, and dropout regularization

3.3 Training and model setup

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). The i-th batch in a given epoch contains the i-th utterance of each dialog of the training set: as a result, the size of the batch varies during the epoch as dialogs have different utterances. For regularization, early stopping is used on the validation set with patience 10.

Our basic setup for the architecture is as follows. For the word embeddings layer, we use the publicly available word2vec vectors that were trained on 100 billion words from Google News[6] (Mikolov et al., 2013a; Mikolov et al., 2013b). For the RNN layer, mono-directional LSTM with output dimension of 500 is used, and for the pooling layer, last pooling is used.

[6] `https://code.google.com/p/word2vec/`

3.4 Experiments on hyperparameters

In addition to comparing with the baselines, we perform a series of experiments on different choices of hyperparameters in order to demonstrate the effect of changing various hyperparameters as follows.

- *Word embeddings*: random initialization vs word2vec pretraining
- *RNN layer*:
  - *Architecture*: LSTM vs GRU
  - *Output dimension*: 100, 250, and 500
  - *Monodirectional* vs *Bidirectional*

- *Pooling*: last, mean, max, and content-based
- *Regularization*: dropout vs no dropout

These hyperparameters are tested one by one, only varying the hyperparameter being tested, while the other hyperparameters remain the same as in the basic setup described in Section 3.3, unless noted otherwise.

4 Results and Discussion

Each experiment involving RNN-based architecture is run 5 times. We report the minimum, average and maximum micro F1-scores across these five runs on the DSTC 4 test set. The experiments on the baseline classifiers are not repeated, since they are deterministic.

Table 2 compares the results for the two baselines (LR and SVM) against the RNN models (LSTM and GRU). The RNN-based architectures improve the performance over the baseline classifiers by almost 10 percentage points. Sequential models show improvements over non-sequential models. Finally, LSTM is slightly better than GRU.

Table 3 shows the performance of monodirectional and bidirectional LSTMs as well as various pooling methods. To match the number of parameters, monodirectional LSTMs use the output dimension of 500, while the bidirectional LSTMs use the output dimension of 250. Among the pooling methods, max pooling outperforms the other methods. In contrast, monodirectional and bidirectional models have comparable performances.

Table 4 presents the results for experiments on different output dimensions of LSTM, random initialization of word vector, and dropout regularization. Smaller LSTM output dimension resulted in slightly better performance, which indicate that small sentence embeddings suffice for this classification task. Randomly initializing the word embeddings yields similar performance to pretraining word embeddings with word2vec. Lastly, regularizing with dropout slightly improves the performance.

5 Conclusion and Future Work

This article has presented a novel approach to sequential short-text classification. Adding context information helps improve the quality of the predictions, and ANN-models outperform traditional approaches. One of the main challenges with ANN models resides in pinpointing the optimal hyperparameters, as some of them may significantly impact the performance.

There is still much room for further investigation. Since unsupervised pre-training of word vectors typically boost the performances of ANNs for NLP tasks, experimenting with other dense word embeddings such as GloVe (Pennington et al., 2014) or Senna (Collobert, 2011; Collobert et al., 2011) may further improve the performance. Using character or morphemes as input instead of word vectors may also help, especially with smaller data sets. Previous works show that character-level inputs work well for language modeling (Kim et al., 2015b) as well as text classification (Zhang and LeCun, 2015).

Different architectures of the ANNs could be experimented. Convolutional Neural Networks (CNNs) are the most obvious choice, as they are a commonly used layer used along with LSTMs and GRUs, and they have been reported to work well for short-text classification (Kim, 2014). It could also be interesting to propose some new variants of existing layers, as the framework we have developed allows fast development iterations.

As the performances may vary depending on the data sets, it would be valuable to compare models across more data sets that could be used for sequential short-text classification annotated with either dialog acts such as the Switchboard Dialog Act Corpus (Jurafsky et al., 1997), the ICSI MRDA corpus (Shriberg et al., 2004) and the NPS Chat corpus (Forsyth and Martell, 2007)), or other type of sequences such as topics.

Acknowledgments

We wish to thank Regina Barzilay and Tommi Jaakkola for teaching this great class, as well as for providing valuable feedbacks during the project meetings.

References

Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP (1)*, pages 1061–1064.

Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

Franck Dernoncourt, Ji Young Lee, Trung H. Bui, and Hung H. Bui. 2016. ADOBE-MIT submission to the DSTC 4 Spoken Language Understanding pilot task. In *7th International Workshop on Spoken Dialogue Systems (IWSDS)*.

Eric N Forsyth and Craig H Martell. 2007. Lexical and discourse analysis of online chat dialog. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 19–26. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dan Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*, pages 97–102.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason Williams, and Matthew Henderson. 2015a. Dialog State Tracking Challenge 4: Handbook.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015b. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason Williams, and Matthew Henderson. 2016. The Fourth Dialog State Tracking Challenge. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

Piroska Lendvai and Jeroen Geertzen. 2007. Token-based chunking of turn-internal dialogue act sequences. In *Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue*, pages 174–181.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics.

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

Norbert Reithinger and Martin Klesen. 1997. Dialogue act classification using language models. In *EuroSpeech*. Citeseer.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. Technical report, DTIC Document.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Daan Verbree, Rutger Rienks, and Dirk Heylen. 2006. Dialogue-act tagging using smart feature selection; results on multiple corpora. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 70–73. IEEE.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.