Dialogue Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network

Hamed Khanpour, Nishitha Guntakandla, and **Rodney Nielsen**
University of North Texas
HiLT Lab
{hamedkhanpour, nishithaguntakandla}@my.unt.edu
rodney.nielsen@unt.edu

Abstract

In this study, we applied a deep LSTM structure to classify dialogue acts (DAs) in open-domain conversations. We found that the word embeddings parameters, dropout regularization, decay rate and number of layers are the parameters that have the largest effect on the final system accuracy. Using the findings of these experiments, we trained a deep LSTM network that outperforms the state-of-the-art on the Switchboard corpus by 3.11%, and MRDA by 2.2%.

1 Introduction

Dialogue Act (DA) classification plays a key role in dialogue interpretation, especially in spontaneous conversation analysis. Dialogue acts are defined as the meaning of each utterance at the illocutionary force level (Austin, 1975). Many applications benefit from the use of automatic dialogue act classification such as dialogue systems, machine translation, Automatic Speech Recognition (ASR), topic identification, and talking avatars (Král and Cerisara, 2012). Due to the complexity of DA classification, most researchers prefer to focus on the task-oriented systems such as restaurant, hotel, or flight, etc. reservation systems.

Almost all standard approaches to classification have been applied in DA classification, from Bayesian Networks (BN) and Hidden Markov Models (HMM) to feed forward Neural Networks, Decision Trees (DT), Support Vector Machines (SVM) and rule-based approaches.

Recently, the advancement of research in deep learning has led to performance upheavals in many Natural Language Processing (NLP) tasks, even leading Manning (2016) to refer to the phenomenon as a neural network "tsunami". One of the main benefits of using deep learning approaches is that they are not as reliant on handcrafted features; instead, they manufacture features automatically from each word (Turian et al., 2010), sentence (Lee and Dernoncourt, 2016; Kim, 2014), or even long texts (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014). Inspired by the performance of recent studies utilizing deep learning for improving DA classification in domain-independent conversations (Ji et al., 2016; Lee and Dernoncourt, 2016; Kalchbrenner and Blunsom, 2013), we propose a model based on a recurrent neural network, LSTM, that benefits from deep layers of networks and pre-trained word embeddings derived from Wikipedia articles.

2 Related Work

Prior work has defined general sets of DAs for domain-independent dialogues that are commonly used in almost all research on DA classification (Jurafsky et al., 1997; Dhillon et al., 2004). The task of DA classification (sometimes called DA identification) is to attribute one member of a predefined DA to each given utterance. Therefore, DA classification is sometimes treated as short-text classification. Similar to many other traditional text classification methods, five sources of information have been used for DA classification tasks: lexical information, syntax, semantics, prosody, and dialogue history. Among all

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2012–2021, Osaka, Japan, December 11-17 2016.

proposed methods, those which used more sophisticated techniques for extracting lexical information, achieved the best results before deep learning was applied to the problem.

DA classification research started with handcrafting lexical features that yielded high quality results with an accuracy of 75.22% on the 18 DAs in the VERMOBIL dataset (Jekat et al., 1995). In general, Bayesian techniques were the most common approaches for DA classification tasks, which used a mixture of n-gram models together with dialogue history for predicting DAs (Grau et al., 2004; Ivanovic, 2005). In some studies, prosody information was integrated with surface-level lexical information to improve accuracy (Stolcke et al., 2000). Stolcke et al. (2000) reported the best accuracy on the core 42 DAs in the Switchboard corpus as 71%. This result was achieved by applying contextual information with HMM for recognizing temporal patterns in lexical information. Novielli and Strapparava (2013) investigated the sentiment load of each DA. They compared the accuracies of the classification before and after analyzing utterances in the Switchboard corpus by using Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2007) and postulated that affective analysis improved the accuracy.

Recently, approaches based on deep learning methods improved many state-of-the-art techniques in NLP including, DA classification accuracy on open-domain conversations (Kalchbrenner and Blunsom, 2013; Ravuri and Stolcke, 2015; Ji et al., 2016; Lee and Dernoncourt, 2016). Kalchbrenner and Blunsom (2013) used a mixture of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs were used to extract local features from each utterance and RNNs were used to create a general view of the whole dialogue. This work improved the state-of-the-art 42-tag DA classification on Switchboard (Stolcke et al., 2000) by 2.9% to reach 73.9% accuracy. Ji et al. (2016) presented a hybrid architecture that merges an RNN language model with a discourse structure that considers relations between two contiguous utterances as a latent variable. This approach improved the result of the state-of-the-art method by about 3% (from 73.9 to 77) when applied on the Switchboard corpus. The best result was achieved when the algorithm was trained to maximize the conditional likelihood. Ji et al. (2016) also investigated the performance of using standard RNN and CNN on DA classification and got the cutting edge results on the MRDA corpus (Ang et al., 2005) using CNN.

3 Our Model

Most deep learning variations were designed and studied in the late 1990s, but their true performance was not revealed until high-speed computers were commercialized and researchers were able to access significant amounts of data. Collobert et al. (2011) used a large amount of unlabeled data to map words to high-dimensional vectors and a Neural Network architecture to generate an internal representation. By adding a CNN architecture Collobert et al. (2011) built the SENNA application that uses representation in language modeling tasks. Their approach outperforms almost all sophisticated traditional NLP applications like part-of-speech-tagging, chunking, named entity recognition, and semantic role labeling without resorting to the use of any handcrafted features or prior knowledge which are usually optimized for each task. In this study, we designed a deep neural network model that benefits from pre-trained word embeddings combined with a variation of the RNN structure for the DA classification task.

For each utterance that contains l number of words, our model convert it into l sequential word vectors. Word vectors can be generated randomly with arbitrary dimensions or being set by a pre-trained word vectors using a variety of word-to-vector techniques (Mikolov et al., 2013; Pennington et al., 2014).

3.1 RNN-based Utterance Representation

Figure 1 illustrates a typical structure of an RNN. As can be seen, information from previous layers, h_{t-1}, is contributed to the succeeding layer's computations that generate h_t. Since almost all tokens, X_i, in a conversation are related to their previous tokens or words, we choose to use an RNN structure.

Given a list of d -dimensional word vectors, $X_1, X_2, ..., X_{t-1}, X_t, ...X_{t+n}$ in a given time step, t, we will have:

$$h_t = \sigma\left(W^{hh} h_{t-1} + W^{hd} X_t\right) \tag{1}$$

$$y_t = softmax\left(W^{(S)} h_t\right) \tag{2}$$

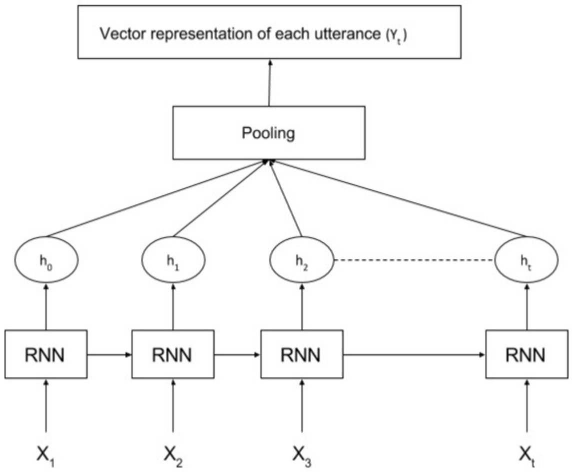

Figure 1: RNN structure for creating a vector-based representation of an utterance from its word.

where $W^{hh}\varepsilon\mathbb{R}^{\mathbb{h}\times\mathbb{h}}$ and $W^{hx}\varepsilon\mathbb{R}^{\mathbb{h}\times d}$ are weight matrices. σ represents logistic sigmoid function, and y_t, $y_t\varepsilon\mathbb{R}^{\mathbb{k}}$, is the class representation of each utterance and k denotes the number of classes for classification task.

In the pooling layer (Figure 1), our model takes all h vectors, $h_{1:t}$, and generate one vector. We can choose from three mechanisms: mean-, max- or last-pooling. Mean-pooling measures the average of all h vectors, max-pooling takes the greatest figure out of each h vector and last-pooling takes the last h vector (i.e., h_t).

Theoretically, RNNs should preserve the memory of previous incidents, but in practice when the gap between relevant information extends, RNNs fail to maintain relevant information. Hochreiter (1991) and Bengio et al. (1994) investigated the main reasons for RNNs' failures in detail. The other problem with RNN is the vanishing and exploding gradient that causes the learning process to be terminated prematurely (Mikolov et al., 2010; Pascanu et al., 2013).

Given the aforementioned problems with RNNs, we use Long Short Term Memory (LSTM), which is a variation of RNNs that is tuned to preserve long-distance dependencies as their default specificity. In DA classification, having the ability to connect related expressions of information that are distant from each other is important, particularly when it comes to classifying utterances as either subjective or objective, which is considered as one of the main sources of error in DA classification (Novielli and Strapparava, 2013). Classifying subjective versus objective texts is one of the major tasks in sentiment analysis in which LSTM-based approaches are shown to achieve high-quality results (Socher et al., 2013). Another reason for using LSTM is that it uses a *forget gate layer* to distill trivial weights, which belong to unimportant words from the cell state (see Eq. 4) . Figure 2 illustrates a standard structure of an LSTM cell.

As can be seen in Figure 2, we can define the LSTM cell at each time step *t* to be a set of vectors in $\mathbb{R}^d$:

$$i_t = \sigma\left(W^{(i)}X_t + U^{(i)}h_{t-1} + b^{(i)}\right) \tag{3}$$

$$f_t = \sigma\left(W^{(f)}X_t + U^{(f)}h_{t-1} + b^{(f)}\right) \tag{4}$$

$$o_t = \sigma\left(W^{(o)}X_t + U^{(o)}h_{t-1} + b^{(o)}\right) \tag{5}$$

$$u_t = \tanh\left(W^{(u)}X_t + U^{(u)}h_{t-1} + b^{(u)}\right) \tag{6}$$

$$c_t = i_t \bigodot u_t + f_{(t)} \bigodot c_{t-1} \tag{7}$$

$$h_t = o_t \bigodot \tanh(c_t) \tag{8}$$

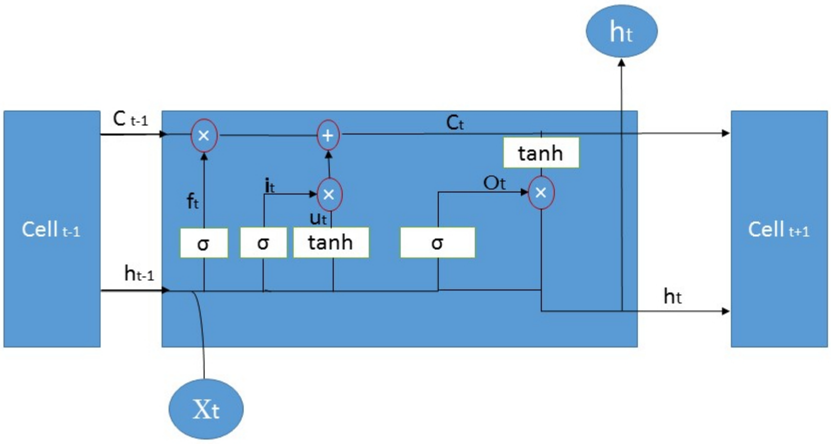

Figure 2: LSTM cell structure and its respective parameters (`http://colah.github.io`).

Where inputs are d dimensional vectors, i_t is the input gate, f_t is the forget gate, o_t is the output gate, c_t is the memory cell, h_t is the hidden state and $\odot$ represents element-wise multiplication.

c_t (Eq. 7) is the key part of LSTMs – it connects chains of cells together with linear interactions. In LSTMs, we have gates in each cell that decide dynamically which signals are allowed to pass through the whole chain. For example, the forget gate f_t (Eq. 4) decides to what extent the previous memory cell should be forgotten, the input gate (Eq. 3) manages the extent to which each cell should be updated, and the output gate manages the exposure of the internal memory state. The hidden layer h_t represents a gated, partial view of its cell state. LSTMs are able to view information over multiple time scales due to the fact that gating variables are assigned different values for each vector element (Tai et al., 2015).

3.2 Stacked LSTM

By arranging some LSTM cells back to back (Figure 2), the hidden layer, h_t, of each cell is considered as input for the subsequent layer in the same time step (Graves et al., 2013; Sutskever et al., 2014). The main reason for stacking LSTM cells is to gain longer dependencies between terms in the input chain of words.

In this study, we used stacked LSTMs with pre-trained word embeddings. Word embedding is distributional representations of words that are used to solve the data sparsity problem (Bengio et al., 2003). We trained word embeddings with 300-dimensional vectors by choosing the window and min-count equal to 5 (Mikolov et al., 2013).

4 Datasets

Since our study focuses on classifying DAs in open-domain conversations, we chose to evaluate our model on Switchboard (SwDA) (Jurafsky et al., 1997) and the five-class version of MRDA (Ang et al., 2005).

- **SwDA:** The Switchboard corpus (Godfrey et al., 1992) contains 1,155 five-minute, spontaneous, open-domain dialogues. Jurafsky et al. (1997) revised and collapsed the original DA tags into 42 DAs, which we use to evaluate our model. SwDA has 19 conversations in its test set.

- **MRDA:** The ICSI Meeting Recorder Dialogue Act corpus was annotated with the DAMSL tagset. This corpus is comprised of recorded multi-party meeting conversations. The MRDA contains 75 one-hour dialogues. There are several variations of the MRDA corpus but MRDA with 5 tags is commonly used in the literature.

  We used the list of files provided by Lee and Dernoncourt (2016) for creating the training, test, and development sets from the MRDA datasets.

5 Experimental Settings

We used the SwDA dataset to tune all hyperparameters including dropout, decay rate, word embeddings and the number of LSTM layers. All conversations in the training set were preprocessed and a randomized selection of one-third of them were utilized as a development set to allow the LSTM parameters to be trained over a reasonable number of epochs. We tuned one parameter value at a time and measured the accuracy on the development set, stopping when the accuracy on the development set did not change for 20 epochs. We used the NN packages provided by Lei et al. (2015) and Barzilay et al. (2016).

5.1 Word Embeddings

We tuned the word embedding parameters $method$, $corpus$ and $dimensionality$, while holding other parameters constant ($dropout = 0.5$, $decayrate = 0.5$ and $layersize = 2$). Specifically, we tested the methods *Word2vec* using the Gensim Word2vec package (Řehůřek and Sojka, 2010) and pretrained *Glove* word embeddings (Pennington et al., 2014). Word2vec embeddings were learned from Google News (Mikolov et al., 2013), and separately, from Wikipedia[1]. The Glove embeddings were pretrained on the 840 billion token Common Crawl corpus.

Method	Resource	Dimension	Accuracy (%)
Word2vec	Wikipedia	75	70.73
Word2vec	Wikipedia	150	71.85
Word2vec	Wikipedia	300	70.77
Word2vec	GoogleNews	75	71.26
Word2vec	GoogleNews	150	71.39
Word2vec	GoogleNews	300	71.32
Glove	CommonCrawl	75	69.28
Glove	CommonCrawl	150	69.71
Glove	CommonCrawl	300	69.40

Table 1: Accuracy using different word embedding techniques, corpora and vector dimensions.

Table 1 illustrates that the best results were consistently achieved by embeddings with 150-dimensions, and of those, Word2vec trained on Wikipedia had the best accuracy. Hence, these settings were used throughout the remainder of the experiments.

5.2 Decay Rate

LSTM uses standard backpropagation to adjust network connection weights (see Eq. 9), where E is the error and W_{ij} is the weight matrix between two nodes, i and j.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}, \tag{9}$$

where η is the learning rate. To avoid overfitting, a regularization factor is added to Eq. 9 to penalize large changes in w_{ij}.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} - \eta \lambda w_{ij}. \tag{10}$$

The term $-\eta\lambda w_{ij}$ is the regularization factor and λ is the decay factor that causes w_{ij} decay in scale to its prior measure. We found that changing η does not impact the accuracy so we set $\eta = 1e - 3$ and change λ to find the best fit for the data (Table 2).

As can be seen from Table 2, the positive trend of increasing accuracy fails after setting $\lambda = 0.8$. Therefore, we set $\lambda = 0.7$ in our experiments.

[1]https://dumps.wikimedia.org/enwiki/20160421

Accuracy (%)	λ
70.76	0.1
70.79	0.2
70.87	0.3
71.32	0.4
71.85	0.5
71.90	0.6
71.95	0.7
70.95	0.8

Table 2: The impact of changing λ on accuracy.

5.3 Dropout

Most of the recent studies that exploit deep learning approaches use the dropout technique (Hinton et al., 2012). Dropout is a kind of regularization technique that prevents the network from overfitting by discarding some weights. In each training cycle, it is possible that some neurons are co-adapted by randomly assigning zero to their weights. Dropout methods were originally introduced for feed-forward and convolutional neural networks but recently have been applied pervasively in the input embeddings layer of recurrent networks including LSTMs (Zaremba et al., 2014; Pachitariu and Sahani, 2013; Bayer et al., 2013). Bayer et al. (2013) report that standard dropout does not work effectively with RNNs due to noise magnification in the recurrent process which results in diminished learning. Since standard dropout is proven not to work effectively for RNNs, we apply the dropout technique proposed by Zaremba et al. (2014) for regularizing RNNs that is used by most studies in the literature employing LSTM models (Lei et al., 2015; Barzilay et al., 2016; Jaech et al., 2016; Swayamdipta et al., 2016; Lu et al., 2016). Zaremba et al. (2014) postulate that their approach reduces overfitting on a variety of tasks, including language modeling, speech recognition, image caption generation, and machine translation. We experimented with dropout probability settings in the range between 0.0 and 0.5.

Accuracy (%)	Dropout probability
71.95	0.5
72.01	0.4
72.05	0.3
72.15	0.2
72.55	0.1
73.29	0.0

Table 3: Impact of changing dropout on accuracy.

As can be seen in Table 3, any dropout at all hurt the accuracy. Hence, the value was set at 0.0 – dropout was not used in later tuning or in the final model.

5.4 Number of LSTM Layers

Finally, we tuned the number of layers. If you utilize only two layers, the model does not detect relevant tokens that are distant from each other. Conversely, if you use too many LSTM layers, the model will be prone to overfitting. We tested values in the range of 2 to 15. Table 4 illustrates our settings' performance on the development set – the accuracy increases up to a 10 LSTM cells before dropping significantly at 15.

5.5 Other Parameters

In addition to the aforementioned parameters, we investigated the impact of changing $L2\text{-}reg$, $pooling$, and $activation$ and finally set them to $1e-5$, $last\ pooling$, and $tanh$ respectively. These settings were

Accuracy (%)	No. of layers
73.29	2
73.61	5
73.92	10
72.90	15

Table 4: Impact of LSTM layers on accuracy.

consistent with previous findings in the literature and we did not observe significant improvements by changing these values.

6 Results and Discussion

In previous sections, we found the best setting for our model, with which we gained the best accuracy on the SwDA development set. In this section, we report our results on the SwDA and MRDA test set.

Model	Accuracy (%)
Our RNN Model	**80.1**
HMM (Stolcke et al., 2000)	71.0
CNN (Lee and Dernoncourt, 2016)	73.1
RCNN (Kalchbrenner and Blunsom, 2013)	73.9
DRLM-joint training (Ji et al., 2016)	74.0
DRLM-conditional training (Ji et al., 2016)	77.0
Tf-idf (baseline)	47.3
Inter-annotator agreement	84.0

Table 5: SwDA dialogue act tagging accuracies.

Table 5 shows the results achieved by our model in comparison with previous works. As a baseline, we consider the accuracy obtained from a Naive Bayes classifier using *tf-idf* bigrams as features (Naive Bayes outperformed other classifiers including SVM and Random Forest). Our model improved results over the state-of-the-art methods and the baseline by 3.11% and 32.85%, respectively.

We also applied our model to classify dialogue acts in the MRDA with 5 dialogue acts. To do so, we used the same settings as described above for classifying dialogue acts in SwDA (Table 5). Table 6 shows our results on the MRDA corpus.

Model	Accuracy (%)
Our RNN Model	**86.8**
CNN (Lee and Dernoncourt, 2016)	84.6
Graphical Model (Ji and Bilmes, 2006)	81.3
Naive Bayes (Lendvai and Geertzen, 2007)	82.0
Tf-idf (baseline)	74.6

Table 6: MRDA dialogue act tagging accuracies.

We calculate the baseline as before, by using *tf-idf* bigram features. The Random Forest classifier achieved the best result in comparison to other classifiers such as Naive Bayes and SVM. Our results in Table 6 show that our model outperformed the state-of-the-art method by 2.2%. It should be emphasized that our model achieved this result without being tuned on an MRDA development set.

7 Conclusion

In this study, we used a deep recurrent neural network for classifying dialogue acts. We showed that our model improved over the state-of-the-art in classifying dialogue act in open-domain conversational text.

We ran several experiments to realize the effects of setting each hyperparameter on the final results. We found that dropout regularization should be applied to LSTM-based structures (even for LSTM-adapted dropout methods that have been proven to have a positive impact on some datasets) cautiously to ensure that it does not have a negative impact on the accuracy of the system.

Acknowledgements

This research is partially supported by grant IIS-1262860 to UNT from the National Science Foundation.

References

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

John Langshaw Austin. 1975. *How to do things with words*. Oxford university press.

Tao Lei Hrishikesh Joshi Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, and Alessandro Moschitti Llu Marquez. 2016. Semi-supervised question retrieval with gated convolutions. *Naacl*.

Justin Bayer, Christian Osendorfer, Daniela Korhammer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. 2013. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

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.

Rajdip Dhillon, Sonali Bhagat, Hannah Carvey, and Elizabeth Shriberg. 2004. Meeting recorder project: Dialog act labeling guide. Technical report, DTIC Document.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Sergio Grau, Emilio Sanchis, Maria Jose Castro, and David Vilar. 2004. Dialogue act classification using a bayesian approach. In *9th Conference Speech and Computer*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, page 91.

Edward Ivanovic. 2005. Dialogue act tagging for instant messaging chat sessions. In *Proceedings of the ACL Student Research Workshop*, pages 79–84. Association for Computational Linguistics.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117*.

Susanne Jekat, Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast, and J Joachim Quantz. 1995. *Dialogue acts in VERBMOBIL*. Citeseer.

Gang Ji and Jeff Bilmes. 2006. Backoff model training using partially observed data: Application to dialog act tagging. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 280–287. Association for Computational Linguistics.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. *arXiv preprint arXiv:1603.01913*.

Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, Van Ess-Dykema, et al. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 88–95. IEEE.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Pavel Král and Christophe Cerisara. 2012. Dialogue act recognition approaches. *Computing and Informatics*, 29(2):227–250.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*.

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.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *arXiv preprint arXiv:1603.00223*.

Christopher D Manning. 2016. Computational linguistics and deep learning. *Computational Linguistics*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

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

Nicole Novielli and Carlo Strapparava. 2013. The role of affect analysis in dialogue act identification. *Affective Computing, IEEE Transactions on*, 4(4):439–451.

Marius Pachitariu and Maneesh Sahani. 2013. Regularization and nonlinearities for neural language models: when are they needed? *arXiv preprint arXiv:1301.5650*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

James W Pennebaker, Roger J Booth, and Martha E Francis. 2007. Linguistic inquiry and word count: Liwc [computer software]. *Austin, TX: liwc. net*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Suman Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. *Proc. Interspeech, Dresden*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Richard Socher, Alex Perelygin, Jean Y 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 conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. *arXiv preprint arXiv:1606.08954*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.