A COMPARATIVE STUDY OF NEURAL NETWORK MODELS FOR LEXICAL INTENT CLASSIFICATION

Suman Ravuri[1,3] Andreas Stolcke[2,1]

[1]International Computer Science Institute [3]University of California, Berkeley, CA, USA
[2]Microsoft Research, Mountain View, CA, USA
ravuri@icsi.berkeley.edu anstolck@microsoft.com

ABSTRACT

Domain and intent classification are critical pre-processing steps for many speech understanding and dialog systems, as it allows for certain types of utterances to be routed to particular subsystems. In previous work, we explored many types of neural network (NN) architectures—some feedforward and some recurrent—for lexical intent classification and found that they improved upon more traditional statistical baselines. In this paper we carry out a more comprehensive comparison of NN models including the recently proposed gated recurrent unit network, for two domain/intent classification tasks. Furthermore, whereas the previous work was confined to relatively small and controlled data sets, we now include experiments based on a large set obtained from the Cortana personal assistant application.

We compare feedforward, recurrent, and gated —such as LSTM and GRU— networks against each other. On both the ATIS intent task and the much larger Cortana domain classification tasks, gated networks outperform recurrent models, which in turn outperform feedforward networks. Also, we compared standard word vector models against a representation which encodes words as sets of character n-grams to mitigate the out-of-vocabulary problem. We find that in nearly all cases, the standard word vectors outperform character-based word representations. Best results are obtained by linearly combining scores from NN models with log likelihood ratios obtained from N-gram language models.

1. INTRODUCTION

Utterance classification is an important pre-processing step for many dialog systems that interpret speech input. For example, a user asking Siri or Cortana to "tell me about the weather" should have her utterance classified as *weather-query* so that the query can be routed to the correct natural understanding subsystem. In previous work [1], we compared feedforward neural network addressee models in a related task of lexical addressee detection, in which a system must identify whether speech is directed at the machine, or another human, and recently, we compared recurrent neural network and long short-term memory units [2] for both addressee and intent detection. Most recently, we have started to investigate gated recurrent units (GRU) models [3] as replacements for LSTMs.

The motivation for neural network models is that previous n-gram-based classification approaches, such as standard LMs and boosting, suffer from two fundamental and competing problems: the limited temporal scope of n-grams, and their sparseness, requiring large amounts of training data for good generalization. The longer the n-grams one chooses to model, the more the sparseness issue is exacerbated. To address sparseness of data, one can try to enlist outside training data for n-gram LMs [4], but these approaches are ultimately limited by the domain-specific nature of n-gram distributions (i.e., models trained from outside data often do not generalize). Neural network language models (NNLMs) [1], are able to enlist outside data to train word embeddings and improve the models.

NNLMs, along with standard n-gram LMs, however, suffer from a different problem: the limited temporal scope of n-grams may render it unable to classify utterances which rely on long-term dependencies ("Tell me about your day" and "how was the Mexican restaurant" are likely directed at other humans while "Tell me about the weather" and "Tell me about Mexican restaurants" are likely directed at the system). Models based on recurrent neural networks (RNN) and long short-term memory (LSTM) [5] certainly outperformed standard n-gram models on small tasks, but how they compare to NNLMs is an open question. Finally, we wanted to investigate word representations based on character n-grams [6] in a large-data setting, since prior comparisons for the task of utterance classification as compared to standard word vectors were inconclusive.

This work aims to compare feedforward, recurrent, and "gated"—such as LSTM and GRU—models on both small, elicited and larger, realistic corpora to determine which models work best under which scenario. Moreover, we would like to compare standard word embedding representations to representations based on character n-grams, which hold the promise of better generalization to unseen words.

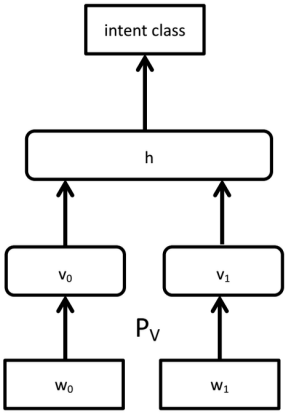

Fig. 1. The standard Neural Network Language Model. P_w, the projection layer, is shared by $\mathbf{w_1}$ and $\mathbf{w_2}$. Word embeddings from those words, v_1, v_2, are stacked into a single vector, which serves as input to a multilayer perceptron. In this architecture, the NNLM is predicting the next word, but can also predict other labels, such as addressee labels.

There is a vast literature on domain and intent classification for purposes of speech understanding; for prior work see [7, 8] and references therein. In this work, we focus on binary classification tasks (which can be treated as detection tasks), although the proposed systems can easily be extended to multi-class situations.

2. COMPARISON SYSTEMS

2.1. Baseline systems

As our experimental baseline we use two classifier architectures based on n-gram features. One system is a pair of class-specific n-gram language models, each of which computes a class likelihood. The log ratio of these likelihoods is then normalized for the utterance length (number of words) to obtain a detection score that is thresholded. A detailed study of this approach can be found in [4]. The other baseline approach is the "Boostexter" boosting algorithm [9, 10], whose output score may also be used as a detection score for thresholding.

2.2. NNLM-based utterance classifier

Figure 1 shows our architecture for the NNLM baseline system, first proposed in [1]. It is based on the Neural Network Language Model, first introduced in [11, 12] as an alternative to the traditional language model. Unlike traditional language models, it encodes words as n-dimensional vectors, with standard dimensions ranging from 100 to 1000 depending on the task. For utterance classification, the NNLM is trained to predict the utterance class based on two consecutive words at a time. The aggregate score for an utterance is calculated as:

$$P(L|\mathbf{w}) \approx P(L_1, \ldots, L_n|\mathbf{w}) = \prod_{i=1}^{n} P(L_i|\mathbf{w})$$
$$\approx \prod_{i=2}^{n} P(L_i|w_{i-2}, w_{i-1}, h_i) = \prod_{i=1}^{n} P(L_i|h_i)$$

The best class is calculated as $\arg\max_L \log p(L|\mathbf{w})$.

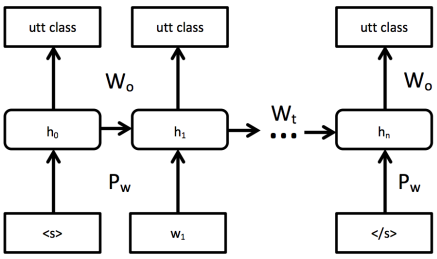

Fig. 2. Proposed RNN classifier model.

2.3. RNN-based utterance classifier

Recurrent neural network language modeling (RNNLM) [13] grew out of the observation that temporal modeling of an entire sentence through a series of hidden units can outperform models based on the Markov (limited memory) assumption. Similar to Neural Network Language Model [11], the RNNLM maps words to a dense n-dimensional word embedding. The hidden state h_t is a function of the current embedding, the previous hidden state, and a bias: $h_t = \sigma(W_t h_{t-1} + v_t + b_h)$. Typically, the optimal dimension for the word embedding is less than half of that in the feedforward language model.

Adapted for use in utterance classification, we train a single RNN model on utterance class labels, shown in Figure 2. The RNN attempts to classify the utterance based on the information stored thus far in h_t. At test time, the probability of an utterance label is calculated as:

$$P(L|\mathbf{w}) \approx P(L_1, \ldots, L_n|\mathbf{w}) = \prod_{i=1}^{n} P(L_i|\mathbf{w})$$
$$\approx \prod_{i=1}^{n} P(L_i|w_i, h_{i-1}) = \prod_{i=1}^{n} P(L_i|h_i)$$

where the final equality is embodied in the softmax output function.

2.4. LSTM- and GRU-based utterance classifier

Ideally a model performing utterance classification would predict a single class label per utterance. In earlier experiments, we did not obtain competitive performance with RNN

models predicting a single label at the end of an utterance, likely due to the vanishing gradient problem. This argues for the use of long short-term memory (LSTM) units for utterance classification.

The LSTM, first described in [5], attempts to circumvent the vanishing gradient problem by separating the memory and output representation, and having each dimension of the current memory unit depending linearly on the memory unit of the previous time step. A popular modification of the LSTM uses three gates—input, forget, and output—to modulate how much of the current, the previous, and output representation should be included in the current time step. Mathematically, it is specified by the equations:

$$\begin{aligned} i_t &= \sigma(W_i v_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f v_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o v_t + U_o h_{t-1} + V_o m_t + b_o) \\ m_t &= i_t \circ tanh(W_c v_t + U_c h_{t-1} + b_c) + f_t \circ m_{t-1} \\ h_t &= o_t \circ tanh(m_t) \end{aligned}$$

where i_t, f_t, and o_t denote the input, forget, and output gates respectively, m_t, the memory unit, and h_t, the hidden state, and is shown in Figure 3. We found that, unlike for RNNs, a model making a single prediction at utterance end does achieve good performance.

More recently, gated recurrent units have been proposed as a simplification of the LSTM [3], while keeping the ability to retain information over long sequences. As in the LSTM, "memory" is handled by a simple linear interpolation between a hidden-like state in the previous time step and a RNN-like component representing a current time step. Unlike the LSTM, however, it uses only two gates, memory units do not exist, and the linear interpolation occurs in the hidden state. We use a slight modification of the original GRU, proposed in [14], as is described by equations:

$$\begin{aligned} z_t &= \sigma(W_z v_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r v_t + U_r h_{t-1} + b_r) \\ h_t &= z_t \circ tanh(W_h v_t + U_h(r_t \circ h_{t-1})) + (1 - z_t) \circ h_{t-1} \end{aligned}$$

One question is whether the one-hot vector w should input directly to the LSTM or GRU, as proposed by [15] for a slot-filling task. We found it better to use a separate linear embedding, and use the embedding v_t as input to the LSTM or GRU. Figure 3 depicts this model.

2.5. Word hashing

One issue we noticed especially with recurrent models is that they are somewhat more sensitive to the occurrence of unknown words than standard feedforward networks. This is best explained by example. Consider an utterance which begins with an unknown word. In a trigram model, only the first two samples are affected by the unknown word, while in recurrent NNs all future hidden states are affected by the unknown word. For corpora with a high percentage of singletons in the training set, this problem is particularly acute, as the standard practice is to map all such words to an unknown token. Moreover, such unknown words may be informative, such as "Kleaners" in "Can you show me the address of Happy Kleaners?". To combat issues with unknown word modeling, we investigate word representations based on sets of character n-grams, as proposed in [6]. A word such as "Kat" is transformed into a set of character n-grams, each of which is associated with a bit in the input encoding. In the case of character trigrams, this hash is the set "#Ka", "Kat", "at#", and the probability of a collision in the hash is less than 0.01%.

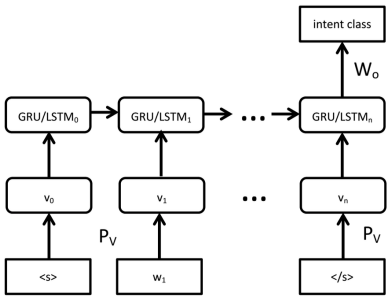

Fig. 3. Proposed LSTM/GRU classifier model.

3. METHOD

For evaluation purposes we chose two very distinct corpora. Our first source of data is ATIS, a small corpus (by today's standards) collected under controlled conditions and carefully transcribed, and widely used in work on automatic speech understanding. Our second corpus, by contrast, is from an actual deployed speech understanding system with open-ended uses and orders of magnitude more data. In both cases, our aim was to define the task such that it was independent of class priors and as similar as possible in nature. As described below, in both cases we defined binary classification task with relatively well-balanced class distributions. As an evaluation metric we use equal error rate (EER), the rate of misclassification obtained when both types of class confusions are equally likely (after choosing an appropriate threshold). This results in a metric that is independent of class priors (the EER of a random classifier, or one that always output the majority class, is 50%).

3.1. ATIS Intent Classification Task

We follow the ATIS corpus setup used in [16, 17] in this paper. The training set comprises 4,978 utterances taken from the Class A (context independent) portions of ATIS-2 and

ATIS-3, and 893 test utterances from the ATIS-3 Nov93 and Dec94 data sets. The corpus has 17 different intents, which we mapped to a binary "flight" versus "other" classification task (70% of the utterances are classified as "flight", though our metric is insensitive to prior distribution, as explained below). Training was based on reference transcripts, but testing used ASR output as described in [8], with a word error rate of around 14%. There are two versions of the input: one uses only the original transcript words, the other—known as "autotagged"—replaces entities by phrase labels such as CITY and AIRLINE, obtained from a tagger [15].

3.2. Cortana Domain Classification

Our second corpus and task is drawn from the Microsoft Cortana personal assistant [7]. Utterances directed at the system need to be routed to different semantic subsystems based on the domain of discourse (such as communication, weather, etc.), and those for which no specialized handling is available are treated as web search queries. The binary classification task we chose for our study is the detection of web search queries versus all others domains. For training purposes, a corpus of 2.1 million utterances was available. Temporally later and disjoint utterance sets were used for development (138k utterances) and testing (221k utterances). Close to half the utterances were in the web search domain (47% of the test set). The inputs to our systems are the words as obtained from automatic speech recognition.

3.3. Experimental Setup

NNLM classifier models use a 500-dimensional word embedding along with a hidden layer of 1000 hidden units, as this was optimal in previous work and new experiments on both data sets. Possibly owing to the size of the word embedding, we found poor results in initial experiments with word hashing; hence they are not included in the study. The RNN, LSTM, and GRU models use a 200-dimensional word embeddings for one-hot and word hashing on both corpora, as those parameters experimentally produced the best results. In addition, the LSTM and GRU included a layer of 15-dimensional hidden and memory units. Not including word embeddings, the LSTM model has roughly 150% more parameters than the RNN, while the GRU uses fewer than the RNN. For large-vocabulary tasks, however, the size of the embeddings dwarf the number of other parameters, so we just use the structure which produced the best results. For word hashes, we use a concurrent trigram and bigram representations. For example, the set describing "cat" is "#c", "ca", "at", "t#", "#ca", "cat", and "at#'.

3.4. NN Training

As noted by other authors, parameter estimation for RNNs is substantially more difficult than for feedforward networks. Well-trained systems typically use a combination of momentum, truncated back-propagation through time (BPTT), regularization, and gradient clipping. Since utterance lengths for the corpora investigated were typically under 20, we found no improvement employing gradient clipping or truncated BPTT. Moreover, regularization had either minimal or deleterious effect. While simple momentum did help for ATIS, more advanced modifications such as Nesterov momentum [18] yielded no improvement, and simple stochastic gradient descent yielded good results on the Cortana corpus.

Despite the relative ease of the intent classification on ATIS, we did find that final results were sensitive to initial parameters, which, for RNN and non-gate LSTM weights were drawn from a $\mathcal{N}(0.0, .04)$ distribution, while LSTM gate weights were drawn from the same distribution, except that the gate biases were a large positive value (around 5) to ensure those values started at approximately 1.0. The variance of performance is investigated in Section 4. We used one heuristic that worked well from prior work [2]: before training, we calculated the cross-entropy on a held-out set across ten random seeds and picked the one which produced the lowest cross-entropy. Variance in results seemed not to be an issue on the Cortana dataset, so we did not include results on the variance of equal error rates.

For ATIS, the initial learning rate for each of the systems is 0.01, with a momentum of 3×10^{-4} for recurrent neural networks, and 3×10^{-5} for LSTM models. The learning rate is halved once the cross-entropy on a held-out set decreases less than 0.01 per example, continues at the same rate until the same stopping point is achieved, and then is halved at each epoch until cross-entropy no longer decreases. As stated earlier, the best initial cross-entropy across 10 different initializations is used.

For the larger Cortana training set, momentum is not needed. The learning rate parameters are similar to that for ATIS, except that the learning rate is decreased when the cross-entropy on a held-out set improve by less than 0.1%. Since the larger dataset included much more training data, initial results suggested that the variance between random initializations is much lower for all models.

For model combination and evaluation, we use linear logistic regression (LLR) to calibrate all model scores and to combine multiple scores where applicable [19]. To estimate LLR parameters on the smaller ATIS dataset, we jack-knife over nine equal-sized partitions of the test data, training on all but one in turn, and cycling through all partitions. Scores are then pooled over the entire test set and evaluated using equal error rate (EER).

For experiments on Cortana data, the development set is used to estimate LLR weights for model combination, which are then evaluated on the test set, also using EER.

Table 1. *ATIS intent classification results. The column labeled autotagged refers to the condition in which certain named entities are marked via lookup table.*

System	EER (%)	Autotag EER (%)
word 3-gram LM	9.37	6.05
word 3-gram boosting	4.47	3.24
NNLM-word	6.05	4.03
RNN-word	5.26	2.45
RNN-hash	5.33	2.81
LSTM-word	**2.45**	1.94
LSTM-hash	2.88	2.81
GRU-word	3.24	**1.58**
GRU-hash	3.24	2.02

4. RESULTS

Table 1 shows results for ATIS intent classification for the different neural network architectures. In both the standard and autotagged setting, the NNLM model is worse than all other models, including a word-trigram boosted model. RNN models are next best, followed by GRU and LSTM models, which performed similarly depending on condition. LSTM models perform better in the regular setting while GRU models perform better in the autotagged setting. Both GRU and LSTM models substantially outperform the other methods, and, as can be seen by Table 2, are within a standard deviation of each other. Hashing seems to perform as well as word vectors on this small dataset.

Table 2 includes mean and standard deviation results for all the models. Feedforward neural networks had by far the lowest standard deviation of all the results, but also had the worst performance. In general, the gated networks had higher variance than recurrent one, although the effect is fairly minimal. Finally, using the best seed for held-out set cross-entropy across ten random seeds is a reasonable method for picking the best random seed.

Results are much clearer for the larger dataset, as shown in Table 3. As with the ATIS data, the relative ordering of models from worst to best is: NNLM, then RNN, and finally LSTM and GRU with roughly equivalent performance. Word hashing seems to perform about as well as word vectors, with it performing worse with gated models, and better for recurrent ones. Keeping character n-grams requires about one third less storage space than the word vectors, as 12k character n-grams are needed while 18k words are needed for this larger dataset.

Figure 4 shows detection error trade-off curves for the two ATIS test conditions and the Cortana task. On the large data set (Cortana) we observe very straight and parallel trade-off curves, meaning that the various systems have no particular strengths or weaknesses on one type of classification error. That also means that the relative ordering of systems with regard to performance stays the same regardless of the chosen operating point.

Table 2. *Average, Best Held Out, and Oracle Errors on ATIS intent classification. Best held out refers to the hypothetical performance when the model with the lowest cross-entropy on a held-out set is chosen (among 10 random seeds) after training.*

System	EER (%)	Autotag EER (%)
Average Error		
NNLM-word	$5.83 \pm .238$	$4.15 \pm .324$
RNN-word	$4.86 \pm .919$	$3.50 \pm .775$
RNN-hash	$4.32 \pm .917$	$2.64 \pm .324$
LSTM-word	$3.38 \pm .986$	2.22 ± 1.01
LSTM-hash	4.05 ± 1.13	2.64 ± 1.02
GRU-word	4.44 ± 1.69	3.58 ± 1.24
GRU-hash	3.79 ± 1.00	2.63 ± 1.08
Best Held-Out X-ent (Oracle Error)		
NNLM-word	6.05 (5.61)	4.03 (3.60)
RNN-word	3.95 (3.95)	2.45 (2.45)
RNN-hash	3.59 (3.24)	2.45 (2.09)
LSTM-word	2.81 (2.45)	2.02 (1.30)
LSTM-hash	3.24 (2.88)	2.02 (1.22)
GRU-word	3.24 (1.70)	1.30 (1.30)
GRU-hash	3.24 (1.30)	2.02 (2.02)

5. CONCLUSIONS AND FUTURE WORK

We have investigated the performance of a variety of neural network architectures for two semantic utterance classification tasks, using both a small, controlled corpus (ATIS) and a large real-life dataset (Cortana). The relative ordering of models according to performance was quite consistent: gated recurrent networks (GRU and LSTM) were best, with roughly equivalent performance, followed by regular recurrent networks, followed by feedforward networks. Gated unit networks performed better than standard n-gram LM-based classifiers or boosting classifiers. However, n-gram LMs were able to further improve the NN-based systems by way of logistic regression combination. Unlike what we found on another utterance classification task (addressee detection) [2], we did not see consistent gains from word-hashing into character n-gram encodings.

The present study only examines lexical information, and what can be inferred from the utterance at hand. In future work it would worthwhile to incorporate nonlexical (e.g., prosodic) information [20], as well as utterance context preceding the one to be classified [7]. With regard to modeling, the inclusion of a layer performing convolution on the word sequence [7] is a promising architectural feature that is orthogonal to the aspects studied here.

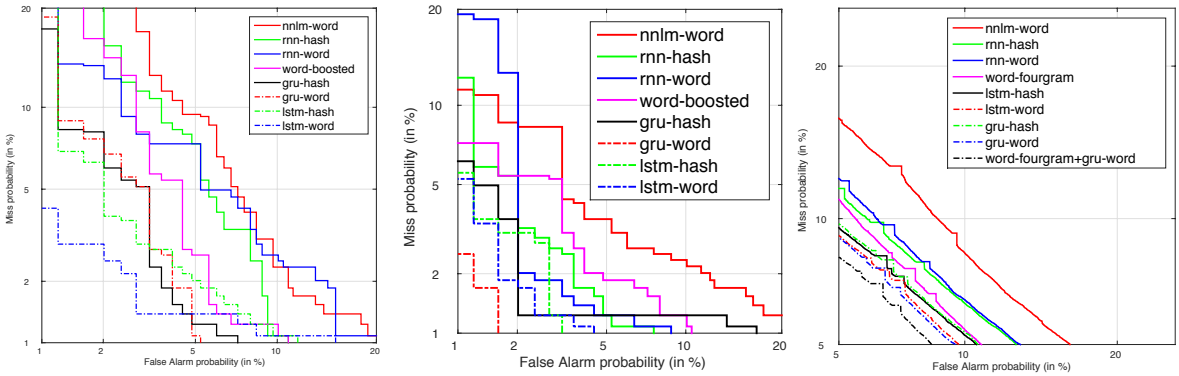

Fig. 4. *DET curves for (a) intent detection from ATIS words (b) intent detection from ATIS autotags and (c) domain classification in Cortana*

Table 3. *Cortana domain classification results. The column labeled "Combo" refers to system performance when NN scores are combined with the baseline word fourgram system. (Note: boosting with 4-grams performed worse than with 3-grams.)*

System	EER (%)	Combo EER (%)
word 3-gram boosting	7.37	–
word 4-gram LM	7.29	–
NNLM-word	9.33	7.30
RNN-word	7.99	6.80
RNN-hash	7.76	6.87
LSTM-word	6.86	6.56
LSTM-hash	7.11	6.63
GRU-word	6.78	6.46
GRU-hash	7.08	6.64

6. ACKNOWLEDGMENTS

We thank Dilek Hakkani-Tur and Kaisheng Yao for their help with setting up the ATIS corpus and discussion on LSTM use for this task. We also wish to thank Minwoo Jeong and Ruhi Sarikaya for assistance with the Cortana data.

7. REFERENCES

[1] Suman Ravuri and Andreas Stolcke, "Neural network models for lexical addressee detection," in *Proc. Interspeech*, Singapore, Sept. 2014, pp. 298–302.

[2] Suman Ravuri and Andreas Stolcke, "Recurrent neural network and LSTM models for lexical utterance classification," in *Proc. Interspeech*, Dresden, Sept. 2015.

[3] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[4] Heeyoung Lee, Andreas Stolcke, and Elizabeth Shriberg, "Using out-of-domain data for lexical addressee detection in human-human-computer dialog," in *Proceedings North American ACL/Human Language Technology Conference*, Atlanta, GA, June 2013, pp. 221–229.

[5] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[6] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck, "Learning deep structured semantic models for web search using clickthrough data," in *ACM International Conference on Information and Knowledge Management*, San Francisco, Oct. 2013.

[7] Puyang Xu and Ruhi Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *Proc. ICASSP*, Florence, May 2014, pp. 136–140.

[8] Gokhan Tur, Dilek Hakkani-Tür, Larry Heck, and S. Parthasarathy, "Sentence simplification for spoken language understanding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*. May 2011, IEEE SPS.

[9] Robert E. Schapire and Yoram Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[10] Benoit Favre, Dilek Hakkani-Tür, and Sébastien Cuendet, "icsiboost. open-source implementation of Boostexter," http://code.google.come/p/icsiboost, 2007.

[11] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," Tech. Rep. 1178, Department of Computer Science and Operations Research, Centre de Recherche Mathématiques, University of Montreal, Montreal, 2000.

[12] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain, "Neural probabilistic language models," in *Studies in Fuzziness and Soft Computing*, vol. 194, pp. 137–186. 2006.

[13] Tomáš Mikolov, Martin Karafiát, Lukaš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, Makuhari, Japan, Sept. 2010, pp. 1045–1048.

[14] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[15] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi, "Spoken language understanding using long short-term memory neural networks," in *IEEE SLT*, 2014.

[16] Yulan He and S. Young, "A data-driven spoken language understanding system.," in *Proceedings IEEE Workshop Automatic Speech Recognition and Understanding*, 2003.

[17] Christian Raymond and Giuseppe Riccardi, "Generative and discriminative algorithms for spoken language understanding.," in *INTERSPEECH*. 2007, pp. 1605–1608, ISCA.

[18] Yu Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$," *Doklady AN SSSR (Soviet. Math. Docl.)*, vol. 269, pp. 543–547, 1983.

[19] Stéphane Pigeon, Pascal Druyts, and Patrick Verlinde, "Applying logistic regression to the fusion of the NIST'99 1-speaker submissions," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 237–248, Jan. 2000.

[20] Elizabeth Shriberg, Andreas Stolcke, and Suman Ravuri, "Addressee detection for dialog systems using temporal and spectral dimensions of speaking style," in *Proc. Interspeech*, Lyon, Aug. 2013, pp. 2559–2563.