SENTIMENT DETECTION FROM SPEECH RECOGNITION OUTPUT

Ivan Tashev[1,2,3*] and Dimitra Emmanouilidou[1]

[1] *Microsoft Research Labs – Redmond, Redmond, WA, USA,*
[2] *University of Washington, Seattle, WA, USA,*
[3] *Technical University of Sofia, Sofia, Bulgaria,*
e-mails: ivantash@microsoft.com; diemmano@microsoft.com

Abstract. Sentiment detection from text has been one of the first text analysis applications. Recently it made serious progress with using deep learning algorithms. Practical use of the sentiment detection includes applications for monitoring the quality of customer service. In this article we perform a review of established and novel features for text analysis, combine them with the latest deep learning algorithms, and evaluate the proposed models. The issues we address are robustness to the low speech recognition rate, the variable length of the text queries, and the case of imbalanced data sets. We use a large labelled dataset from real support calls, and propose new optimality criterion, which is a combination of weighted and unweighted accuracy.

Keywords: sentiment detection, text analysis, deep learning.

1. INTRODUCTION

Affective computing [1] is the art of recognizing emotions from various modalities. It is widely growing within the field of Human Computer Interaction where speech remains a primary form of expressive communication. Predominantly, speech emotion recognition systems are built to classify speech utterances, which comprise of one dialog turn and typically range a few seconds in duration, or 5–10 words in length. It is assumed that there is one emotion in each utterance and the classification can be either categorical: into discrete categories such as sadness, anger, happiness, neutral [2]; or continuous: emotional attributes such as arousal (passive vs active), and valence (positive vs negative) [3]. For analysis of customer service telephone calls the classification happens only on the valence axis (positive, neutral, negative).

*Corresponding author.

DOI: 10.7546/EngSci.LVII.20.02.01

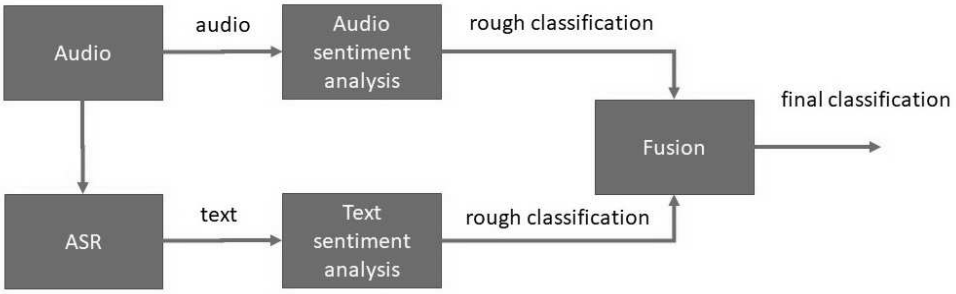


Fig. 1. Typical architecture of a sentiment detection framework

Each classification is performed on one dialog turn, i.e. one utterance of customer speech. Classifications from each of the utterances in the call are later fused to form the final evaluation of the customer call. Typical architecture of such multi-modal sentiment classification system from utterances is shown in Fig. 1. The input audio utterance is sent to an Audio Sentiment Analysis block, and to an Automatic Speech Recognition (ASR) block. The recognized text is processed by a Text Sentiment Analysis block. The conclusions of these two classifiers are fused and form the final classification. In this article we explore various features and classifiers for sentiment analysis based on a single utterance of a customer call. It is extended version of our conference paper [4].

For processing the audio (Audio sentiment analysis), Deep Neural Networks (DNNs) [5] are frequently used. The simplest DNN systems for emotion recognition are feedforward networks of fully connected (FC) layers that are built on top of the utterance level feature representations [6]. Recurrent Neural Networks (RNNs) [7] are a class of neural networks that have cyclic connections between nodes in the same layer. These networks capture the inherent temporal context in emotions and have shown improved performance for classification task [8]. Another class of DNNs, Convolutional Neural Networks (CNNs) [9], capture locally present context, patterns, working on frame level features. CNNs enable the training of end-to-end systems where the feature representations and classification are trained together using a single optimization [10]. Algorithms for sentiment detection from speech are out of the scope of this article, but we will reuse some of the neural network architectures.

Emotion and sentiment detection from text (Text sentiment analysis) is one of the first applications of text analysis. Initial papers were rule-based algorithms, later replaced by bag of words (BoW) modeling using a large sen-

timent or emotion lexicon [11], or statistical approaches that also assume the availability of a large dataset annotated with polarity or emotion labels [12]. Word embedding [13] emerged as a powerful tool to map words with similar meaning closer together. It also can be used to transfer the knowledge from large numbers of unlabeled documents [14] to smaller labeled datasets, in the context of sentiment analysis.

Analysis of text utterances using deep neural networks faces the problem of different number of words in the utterance, while classifiers (SVM, FC DNNs) expect fixed number of input features. One approach is to extract utterance statistics based on the word features a priori, and use the extracted statistics as input to the classifier; alternatively, statistics can be extracted after individual word classification and then combined into a final decision. A third approach is to use models with an intermediate hold state, such as Hidden Markov Models (HMMs) or RNN.

Sentiment analysis from call center conversations faces additional set of problems: the noise in the audio signal that harms both the audio-based classification and the ASR; the need for speech diarization into customer and agent speech; the need for robust text classifiers that overcome inevitable ASR errors. Another aspect of the sentiment classification from audio and text in real-life customer calls is that the collected and labeled datasets are highly imbalanced, with neutral label dominating – typically above 90%. If we train the classier on weighted accuracy (WA) we will have very poor results for the positive and negative classes. Even worse, a classifier which outputs always neutral already will have above 90% WA. If we train the classifier on unweighted accuracy (UA) instead, then we will end up with a high absolute number of neutral phrases misclassified as positive or negative, which is also non-ideal.

In this article we explore various features and classifiers for sentiment detection from the output of ASR from real-life customer service calls. To address the issue with the imbalanced dataset we propose a new cost function to train the classifiers, which is a weighted sum of UA and WA. The article is structured as follows. In Section 2 we describe the real-life dataset and the approaches for labeling it. Section 3 covers the investigated feature sets, Section 4 – the classifier architectures. We provide the experimental results in Section 5 and we finish the article with discussion of the results and draw some conclusions in Section 6.

2. DATASET AND EVALUATION

The dataset is created from recorded Microsoft customer support calls for a range of products and services. It consists of 1957 sessions in total. Each conversation has been automatically segmented into utterances and separated into agent and customer speech (although occasional mix-ups occur due to crosstalk or processing glitches). An initial transcription pass is done automatically, followed by a human transcription. For the purposes of our task, we will use only the audio data from the customer side, with initial number of 139,493 utterances.

2.1. Labelling

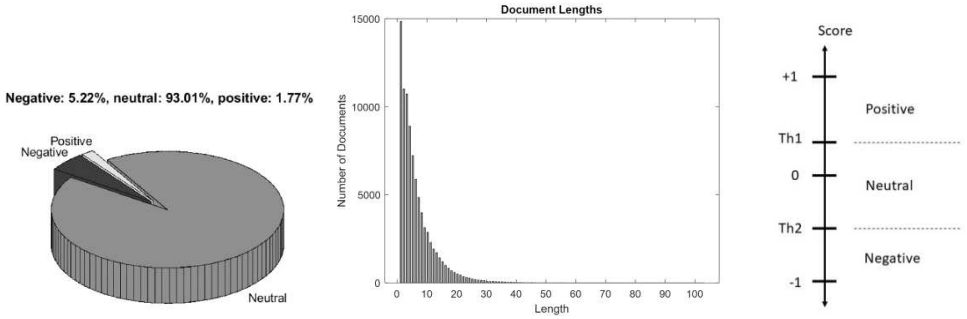
Each utterance is labeled for sentiment by three judges in the Microsoft UHRS crowd-sourcing system. All judges must pass a qualifying test, scoring at least 75% on "gold set" of pre-labeled utterances. Judges listen to the entire conversation, one utterance at a time. Additionally, human-transcribed text is presented on-screen for both current and context utterances (three previous and three following). The context displayed includes both agent and customer utterance. Each judge labels the utterance using one of the following labels: clearly positive, somewhat positive, neutral, somewhat negative, clearly negative, agent speech, not intended for service (side talk), cannot label.

2.2. Data selection

The data selection includes removing all utterances labeled agent speech, not intended for service, cannot label; collapsing somewhat and clearly labels together; leaving only the utterances where at least two of the judges agree. In the final dataset we have 111,665 utterances left, with three labels: positive, neutral, and negative. For each utterance we have noisy transcription (the output of ASR) and exact transcription.

2.3. Dataset analysis

The overall judges' agreement leads to UA of 84.85%. The labels distribution is 93.01% neutral, 5.22% negative, and 1.77% positive as shown in Fig. 2(a). The utterances contained between 1 and 97 words, where 95% of them contained less than 18 words. The histogram of the utterances' length is shown in Fig. 2(b). More detailed analysis of the judges' performance and the dataset can be found in [15]. The dataset was split on training, validation, and testing sets in proportion 80%–10%–10%.



(a) (b) (c)

Fig. 2. Dataset statistics and conversion from score to class: (a) distribution of the labels; (b) number of words in utterance; (c) conversion from score to class

2.4. Evaluation parameters and cost function

Class labels were assigned in a way that emphasizes natural proximity between pairs of classes: -1 for negative, 0 for neutral, and +1 for positive. This way the negative class is closer to neutral than positive. All classifiers initially act as regressors that estimate one score value; the score value is then converted to a class membership using two thresholds as shown in Fig. 2(c).

The first evaluation parameter for the classifier is weighted accuracy (WA):

$$WA = \frac{CL}{N}, \tag{1}$$

where WA is the weighted accuracy, CL is the total number of correct labels, and N is the total number of labels. Note that a classifier that always returns neutral achieves 93% WA on the imbalanced dataset.

The second evaluation parameter is unweighted accuracy (UA):

$$UA = \frac{1}{K} \sum_{k=1}^{K} \frac{CL_k}{N_k}, \tag{2}$$

where UA is the unweighted accuracy, N_k is the total number of labels in class $k = 1, \ldots, K$ and CL_k is the total number of correct labels in class k. Given the three classes of the dataset, a classifier that always returns neutral achieves 33% UA.

With WA as a cost function during training, the trained neural network will tend to return mostly neutral, reducing the accuracy for the other two classes. With UA as a cost function, we will have a very large absolute number of class

neutral classified as one of the other two classes. This will make the manual investigation of customer support calls more difficult and time consuming. To address this issue, we propose using as a cost function the weighted sum of the two accuracies:

$$Q = \alpha WA + (1 - \alpha) UA - 0.001 T_t, \tag{3}$$

where Q is the cost function, WA and UA are the weighted and unweighted accuracies, in %, coefficient α denotes the tradeoff between UA and WA, and T_t is the classifier training time in seconds. The last member is a protection against classifiers with very long training time and minimal advantage in accuracy.

The thresholds in Fig. 2(c) are determined as follows:

$$[Th1, Th2] = \underset{Th1,Th2}{\arg\max} \left(Q_{val}\right), \tag{4}$$

where $Th1$ and $Th2$ are the thresholds, Q_{val} is the cost function on the validation set.

3. FEATURES

The input for the feature extractor is a sequence of words with variable length. The goal is to provide the classifier with the most informative for the task set of features.

3.1. Statistical features

In this group are the classic for the field of computational linguistics n-grams: uni-grams, bi-grams, and tri-grams [16]. Each n-gram is represented as one-hot vector and we let the classifier learn which n-gram is carrying more information about the utterance sentiment of a given class. The feature set for the utterance is the sum of all one-hot vectors.

This feature set can be further augmented with information about the frequency of the n-grams in the utterances of each class, which in information retrieval is called TF*IDF (term frequency–inverse document frequency) [17]. In this case each n-gram is represented by a sparse vector with length the number of n-grams in each class and as many different than zero numbers as classes we have, containing the TF*IDF number of the n-gram for each class. Illustration of the frequency of different words in different classes is shown in the word clouds in Fig. 3.

Fig. 3. Word clouds for the three classes in the training set: negative (left), neutral (center) and positive (right)

One of the problems in the statistical features is out-of-vocabulary (OOV) n-grams or TF*IDFs, which are not presented in the training set, but seen in the test and/or validation dataset. Both n-grams and TF*IDF features are frequently referred to as bag-of-words (BoW) features as they do not keep track of the sequence, i.e. the position of the n-gram in the utterance is not accounted for.

3.2. Embedded features

Word embedding represents each word as a long vector, i.e. as a point in a large dimensional space. Because of the way this vector is derived [13] the words with similar meaning are close together. Even more, in this space king-man+woman is very close to queen. For sentiment detection from text we can use embedded vectors pre-trained on a large data corpus, such as the 16 billion documents dataset in [14]. The probability of OOV words will generally be small, but the word embedding is language dependent and will not be domain specific.

A second approach is to train the word embedding on the words in the training set. In this case the embedded space will be domain-specific, but we can have increased number of OOV words in the validation and test datasets.

Third approach is to train the embedding jointly with the sentiment classifier. Then in the embedded space words informative for a given class will be closer together. The problem with OOV words will still be present.

We can use the word embedding vectors as a sequence, or compute statistics across all the word embeddings in the utterance: mean, max, min, standard deviation.

4. CLASSIFIERS

Most of the classifiers expect fixed length input, while the number of words in a customer utterance can vary significantly. This means that we either have to do some statistical processing of the features before the classifier, or to do classification of each word and then do statistical processing of the outputs, or use classifiers that carry a state from word to word and output the final conclusion at the end of the utterance. For sentiment detection the output of the classifier is a score, which is converted to class decision as described in Section 2.4.

4.1. Fixed input length

The potential feature sets for these classifiers is the BoW group. There are multiple classifiers, described in the literature, but in this article we limit the scope to two: Extreme Learning Machine (ELM) [18] and feed forward fully connected (FC) neural network frequently referred to simply as DNN [5]. The first is simple to train and slightly outperforms on the task all the traditional classifiers like support vector machines (SVM). The second classifier can be made enough deep to have substantial abstraction power and the performance usually is limited by the size of the available dataset.

4.2. Classifiers with state

In this group we experiment with Long-Short Term Memory (LSTM) [19] classifiers, which are in the group of RNN. The LSTM classifiers process the input features consecutively and account for the order of the input vectors. They also preserve internal state and output the decision at the end of the input sequence. LSTM classifiers perform better than the traditional HMM [20]. In addition, an LSTM-based neural network can have more than one LSTM layer.

4.3. Hyper-parameters optimization

Each neural network has hyper-parameters, describing the architecture: number of layers, number of neurons in each layer, etc. The classification accuracy depends on these hyper-parameters and on the dataset. All of the results in this article are presented after a formal process of hyper-parameters optimization, using the cost function, defined in equation (3), as optimization

criterion. The optimization space is small, and the optimization is carried iteratively, one hyper-parameter at a time. This is known as Gaussian minimization. For each of the single dimensional optimization procedures a scanning method is used with eventual quadratic interpolation.

5. EXPERIMENTAL RESULTS

We present the results in three main parts: using BoW as features, using word embedding as features, and exploring the effect on the speech recognition errors. All the classifiers are implemented in MATLAB using the Text Analytics toolbox. Criterion (3) is used for optimization, where $a = 0.5$. The training times, mentioned in the results, are measured on a Windows computer with 12-core, 3.6 GHz, 64-bits CPU and 128 Gbytes of RAM. The GPU is NVIDIA GeForce GTX980Ti. All the design decisions and algorithmic performance ranking were based on performance achieved on the validation dataset. The performance numbers from the test set are provided as evidence for the generalization of the proposed approaches.

5.1. Using bag of words as features

The numerical results from various neural networks using BoW features are provided in Table 1. We use as features uni-grams, bi-grams, tri-grams and all of them simultaneously. A separate experiment is done using TF*IDF features. The two neural networks we experimented with are ELM and DNN. Column Notes describes the architecture of the neural network: the number of hidden layers (hl) and the number of neurons in each layer (hu). As expected, the fully connected deep neural network performs better than ELM with its single hidden layer. The combined feature set of 1-, 2-, and 3-grams provides the highest performance on the validation dataset, achieving $Q = 75.63$. This is the model that achieves the highest $UA = 64.22\%$. Worth mentioning the performance of the same neural network using uni-grams with $Q = 75.08$ and the ELM performance with uni-grams as features and $Q = 74.80$. The good performance of the uni-grams as feature can be explained with the lower number of OOV uni-grams, compared to bi-grams and tri-grams. It is reasonable to expect that with larger datasets the combined use of all three features will perform even better. The results from the test set confirm the good generalization and have similar ranking.

Table 1. Results for bag-of-words as features

Classifier	Features	Validation set				Notes	Test set		
		WA,%	UA,%	T, sec	Q		WA,%	UA,%	Q
ELM	unigrams	87.89	61.74	18.5	**74.80**	1500 hu	87.89	60.89	74.39
	bigrams	91.04	45.92	159.2	68.32	6620 hu	92.02	45.94	68.98
	trigrams	92.56	35.09	3.3	63.82	2000 hu	93.23	34.75	63.99
	1-, 2-, and 3-grams	88.15	58.78	121.0	73.34	4000 hu	89.37	59.20	74.29
	TF*IDF	89.68	56.61	27.5	73.11	2000 hu	90.13	58.26	74.20
DNN	unigrams	88.33	62.49	335.0	**75.08**	3 hl, 800 hu	89.48	64.92	77.20
	bigrams	91.72	46.89	843.0	68.46	4 hl, 1024 hu	92.85	46.12	69.49
	trigrams	92.50	35.33	8.5	63.91	2 hl, 128 hu	93.31	34.91	64.11
	1-, 2-, and 3-grams	87.35	**64.22**	151.6	**75.63**	3 hl, 512 hu	88.04	66.23	77.14
	TF*IDF	89.44	60.33	237.7	74.65	4 hl, 256 hu	90.40	62.76	76.58

Table 2. Results for word embedding as features

Embedding	Features	Classifier	Validation set				Notes	Test set		
			WA,%	UA,%	T, sec	Q		WA,%	UA,%	Q
Pre-trained on 16B words	stats: # words, mean min, max, stdev	ELM	88.67	56.23	169.0	72.28	300 emb,5265 hu	89.68	59.06	74.37
		DNN	84.48	63.89	530.0	73.66	300 emb,3 hl,600 hu	85.79	65.91	75.85
	embeddings as a sequence	LSTM+FC, last	88.16	**62.18**	233.0	**74.94**	300 emb,150 hu	89.17	63.47	76.32
		LSTM+stats+ELM	88.80	61.18	291.0	74.70	300 emb,480 hu	89.76	63.08	76.42
Pre-trained on dataset	train set	LSTM+FC, last	89.27	57.59	752.0	72.68	300 emb,256 hu	89.88	57.23	73.56
	train+val sets	LSTM+FC, last	89.23	58.04	747.0	72.89	300 emb,256 hu	89.91	58.55	74.23
Embedding trained jointly		LSTM+FC, last	89.64	**59.77**	265.0	**74.44**	200 emb,180 hu	90.27	58.43	74.35

5.2. Using embedding as features

Table 2 shows the results from the experiments with word embedding. Again, the Notes column gives information about the neural network architecture.

The first group of experiments uses a pre-trained word embedding on 16 billion documents in American English. This drastically reduces OOV words. As the utterances have different length, in one of the cases we take statistics of the embedding vectors with length of 300: mean, max, min, standard deviation and add as additional feature the number of the words in the utterance. These 1201 features from each utterance are the input of two fully connected neural networks: DNN and ELM. Another approach is to treat the embedding vectors as a sequence of 300 features and use an LSTM network as a classifier. In one of the cases we have an additional fully connected layer at the LSTM output, in another we collect the LSTM outputs after each word. In the second case at the end of the utterances we do statistics as above (mean, min, max, standard deviation, number of words) and finalize the decision using ELM neural network. These two approaches perform well, with slight advantage of the classic LSTM and FC after the output. It achieves $Q = 74.94$ and $UA = 62.18\%$.

A second group of experiments is with word embedding trained either on the training set or trained on the joint training+validation sets. The advantages here are that the embeddings are domain specific, the disadvantages – the dataset is small (less than a million words). The best performing classifier from the previous group of experiments was used (LSTM+FC, last), but in general this group has lower results than the first one.

The last experiment is to train the classifier and embedding jointly. In this case the embedding vector caries information about how much this word belongs to a given class. The used classifier is again LSTM+FC and this is the third best performing configuration using word embedding. It does not require a language specific pre-trained word embedding and does not depend on the quality of such pre-trained embedding. The price for this is minimal hit in the performance, which can be mitigated with a larger dataset. From this standpoint the third approach is the winner of the classifiers using embedding as features.

5.3. Impact of the speech recognition errors

The impact of speech recognition accuracy is explored in Table 3. We perform training and evaluation of the three best performing algorithms using: (i) the ASR output, and (ii) the exact human-written transcription. For

Table 3. Impact of the speech recognition errors

Dataset	Classifier	Features	Validation set				Notes	Test set			Delta, %
			WA,%	UA,%	T, sec	Q		WA,%	UA,%	Q	
ASR output	LSTM+FC	embedding	89.64	59.77	265.0	74.44	200 emb, 180 hu	90.27	58.43	74.35	
	DNN	unigrams	88.33	62.49	335.0	75.08	3 hl, 800 hu	89.48	64.92	77.20	
	DNN	1-,2-,3-grams	87.35	64.22	151.6	**75.63**	3 hl, 512 hu	88.04	66.23	77.14	
Exact transcription	LSTM+FC	embedding	87.98	70.99	533.1	78.95	200 emb, 180 hu	88.53	67.00	77.77	4.51
	DNN	unigrams	88.97	72.74	247.7	**80.61**	3 hl, 800 hu	89.87	73.49	81.68	5.53
	DNN	1-,2-,3-grams	87.57	**73.51**	521.9	80.02	3 hl, 512 hu	88.35	73.86	81.11	4.38

convenience, the results using the ASR output are copied from the previous two tables. The Delta column shows the increase of the performance on the verification set when using the exact transcription. Using the exact transcription shows an expected increase in accuracy, between 4–6%. The highest performance of $Q = 80.61$ is achieved with the combination uni-grams and DNN, while using all n-grams with DNN achieves highest $Q = 73.51$. Note that for clear comparison we use the network architectures, optimized for the ASR output. It is highly probable that an optimization of the neural network architectures for the exact transcription will achieve even higher performance.

6. DISCUSSION AND CONCLUSIONS

In this article we explored various feature representations and classifiers for the task of sentiment detection from speech transcription. We proposed the use of a cost function that accounts for both WA and UA via equation (3), aiming to mitigate highly imbalanced training dataset. A tradeoff coefficient of $a = 0.5$ was proposed. Figure 4 shows the dependency of the WA and UA on this parameter, for DNN classifier using all n-grams as features. The trends for other combinations of features and classifier are quite similar. When $a = 1.0$ we have $WA = 94\%$ and $UA = 42\%$. When the coefficient is moved to the other extreme, $a = 0.0$, then weighted accuracy goes down to 67% and unweighted accuracy goes up to 69%. Seems that using $a = 0.5$ provides a good tradeoff between the weighted and unweighted accuracy, where we have most of the gain in unweighted accuracy (up to 64%), without losing much from weighted accuracy (down to 87%).

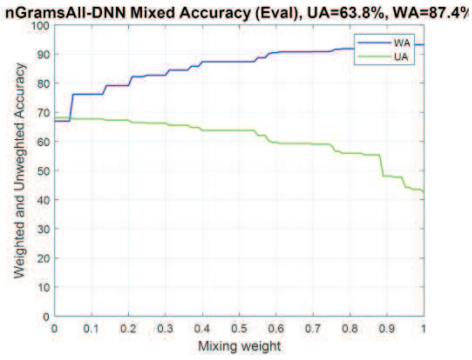


Fig. 4. Weighted and unweighted accuracy dependency on the cost

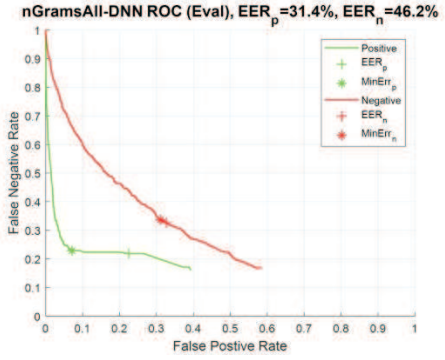


Fig. 5. ROC curves per class

Another aspect of the proposed algorithm is to use the classifiers in regression mode and estimate a value ranging from -1 (negative), through 0 (neutral), to $+1$ (positive). This introduces the concept that negative is closer to neutral than to positive, but also allows applying of two separate thresholds to adjust individually the false positive and false negative rates for the negative and positive classes. The ROC curves for these two classes are shown in Fig. 5. It is well visible that the classification error is much lower for the positive class, than for the negative class. Customers are typically polite and express verbally when they are satisfied but can be less clear in their words when the result is not satisfactory.

In terms of robustness to ASR errors, the analysis of various features and network architectures revealed that 1-, 2-, and 3-grams as features and a DNN-based classifier were the best choice. It is closely followed by the same classifier using just uni-grams, and using pre-trained word embedding as a sequence and LSTM classifier.

In our experiments the traditional classifiers are represented by the ELM, which performs close, but better than pretty much all of them. The proposed algorithms outperform the ELM based classifier with 5–7% in UA for word embedding features and 2–6% in UA for n-gram features.

The logical next step is to combine the results from both audio-based and text-based classifiers. Since the feature rate is different between audio (frames every 20 ms) and text (words on the output of ASR), a late fusion of the two models would be suitable for combining the outputs into a single decision.

ACKNOWLEDGEMENTS

Authors would like to thank our colleagues Ashley Chang, Bryan Li, Dimitrios Dimitriadis, and Andreas Stolcke for labeling the data and fruitful discussions on various aspects of sentiment detection from customers' calls.

REFERENCES

[1] S. Poria, E. Cambria, R. Bajpai, and A. Hussain, A review of affective computing: From unimodal analysis to multimodal fusion, *Information Fusion* (2017) **37** 98–125.

[2] B. Schuller, A. Batliner, S. Steidl, and D. Seppi, Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first chalenge, *Speech Communication* (2011) **53** (9) 1062–1087.

[3] F. Weninger, F. Ringeval, E. Marchi, and B. Schuller, Discriminatively trained recurrent neural network for continuous dimensional emotion recognition from audio, in Proceedings of IJCAI, 2016, pp. 2196–2202.

[4] I. Tashev and D. Emmanouilidou, Sentiment detection from ASR output, in: International Conference on Information Technologies, IEEE, September 2019, pp. 65–68.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, A fast learning algorithm for deep belief nets, *Neural Computation* (2006) **18** (7) 1527–1554.

[6] K. Han, D. Yu, and I. J. Tashev, Speech emotion recognition using deep neural network and extreme learning machine, in: Interspeech 2014.

[7] G. Keren and B. Schuller, Convolutional RNN: an enhanced model for extracting features from sequential data, *Preprint arXiv:1602.05875*, 2016.

[8] J. Lee and I. J. Tashev, High-level feature representation using recurrent neural network for speech emotion recognition, in: Interspeech 2015.

[9] Ya. LeCun, Fu-Jie Huang, and L. Bottou, Learning methods for generic object recognition with invariance to pose and lighting, in: Computer Vision and Pattern Recognition Conference, 2004.

[10] S. Parthasarathy and I. J. Tashev, Convolutional neural netwok techniques for speech emotion recognition, in: Proceedings of IWAENC, September 2018.

[11] G. Mishne *et al.*, Experiments with mood classification in blog posts, in: Proceedings of ACM SIGIR 2005 Workshop on Stylistic Analysis of Text for Information Access, 2005, pp. 321–327.

[12] L. Oneto, F. Bisio, E. Cambria, and D. Anguita, Statistical learning theory and ELM for big social data analysis, *IEEE Comput. Intell. Mag.* (2016) **11** (3) 45–55.

[13] T. Mikolov, I. Sutskever, Kai Chen, G. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of NIPS, 2013.

[14] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, Advances in Pre-Training Distributed Word Representations, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[15] B. Li, D. Dimitriadis, and A. Stolcke, Acoustic and Lexical Sentiment Analysis for Customer Service Calls, in: Proceedings of ICASSP, IEEE, May 2019.

[16] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, Class-based n-gram models of natural language, *Computational Linguistics* (1992) **18** 467–479.

[17] K. Sparck Jones, A statistical interpretation of term specificity and its application in retrieval, *Journal of Documentation* (1972) **28** 11–21.

[18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* (2006) **70** (1) 489–501.

[19] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation* (1997) **9** (8) 1735–1780.

[20] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE* (1989) **77** (2) 257–286.

Received April 18, 2020