Proceedings of the 2019 IEEE International Conference on Information Technologies (InfoTech-2019)
19-20 September 2019, St. St. Constantine and Elena, Bulgaria

Sentiment Detection from ASR Output

Ivan J. Tashev, IEEE Senior Member, Dimitra Emmanouilidou, IEEE Member
Microsoft Research Labs
One Microsoft Way, Redmond, WA, USA
{ivantash, diemmano }@microsoft.com

Abstract – **Emotion and sentiment detection from text have been one of the first text analysis applications. Practical use includes human-computer interaction, media content discovery and applications for monitoring the quality of customer service calls. In this paper 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 for the needs of sentiment detection for monitoring of the customer satisfaction from support calls. The issues we address are robustness to the low ASR recognition rate, the variable length of the text queries, and the case of highly imbalanced data sets. The proposed approaches are shown to significantly outperform the accuracy of the baseline algorithms.**

Keywords – **deep learning; sentiment detection; text analysis.**

I. INTRODUCTION

Affective computing [1] is the art of recognizing emotions from various modalities. It is widely growing within the field of Human Computer Interaction (HCI) 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). Classifications from each of the utterances in the call are later fused to form the final evaluation of the customer call. This paper covers sentiment recognition from the recognized by an Automatic Speech Recognition (ASR) block text, and we explore various features and classifiers for sentiment analysis based on a single utterance of a customer call.

Emotion and sentiment detection from text 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 sentiment or emotion lexicon [4], or statistical approaches that also assume the availability of a large dataset annotated with polarity or emotion labels [5]. Word embedding [6] 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 [7] to smaller labeled data sets, in the context of emotion or sentiment analysis.

©2018 IEEE

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 (HMM) 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 data sets 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. 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 paper 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 paper is structured as follows. In section II we describe the real-life data set and the approaches for labeling it. Section III covers the investigated feature sets, section IV – the classifier architectures. We provide the experimental results in section V and we finish the paper with discussion of the results and draw some conclusions in section VI.

II. DATASET AND EVALUATION

The dataset is created from recorded Microsoft customer support calls, and 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 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.

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), *can't label*.

The data selection includes removing all utterances labeled *agent speech*, *not intended for service*, *can't 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.

The overall judges' agreement leads to UA of 84.85%. The labels distribution is 93.01% *neutral*, 5.22% *negative*, and 1.77% *positive*. The utterances contained between 1 and 97 words, where 95% of them contained less than 18 words. More detailed analysis of the judges' performance and the dataset can be found in [8]. The dataset was split on training, validation, and testing sets in proportion 80%–10%–10%.

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.

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, Nk is the total number of labels in class k=1,...,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* misclassified 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:

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

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 are determined as follows:

$$[Th1, Th2] = \arg\max_{Th1,Th2}(Q_{val}) \tag{4}$$

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

III. 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.

In the group of the statistical features are the classic for the field of computational linguistics n-grams: unigrams, bigrams, and trigrams [9]. 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) [10]. 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.

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.

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 [6] the words with similar meaning are close together. Even more, in this space *king-man+woman* is very close to *queen*. For sentiment detection form text, we can use embedded vectors pre-trained on a large data corpus, such as the 16 billion documents dataset in [7]. 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 data sets.

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 embedding in the utterance: mean, max, min, standard deviation.

IV. CLASSIFIERS

Most of the classifiers expect fixed input length, while the number of words in the utterance varies. 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.

The potential feature sets for the classifiers with fixed input length is the BoW group. In this paper we limit the scope to two: Extreme Learning Machine (ELM) [11] and feed forward fully connected (FC) neural network frequently referred to simply as DNN [12].

In the group of classifiers with state we experiment with Long-Short Term Memory (LSTM) [13] 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 [14].

Each neural network has hyper-parameters, describing the architecture: number of layers, number of neurons in each layer, etc. All of the results in this paper 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. For each of the single dimensional optimization procedures a scanning method is used with eventual quadratic interpolation.

V. EXPERIMENTAL RESULTS

All the classifiers are implemented in MATLAB using the Text Analytics toolbox. Criterion (3) is used for optimization, where $\alpha = 0.5$. The training times 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 are 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.

The results from using BoW as features are provided in Table I. We use as features unigrams, bigrams, trigrams 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 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 unigrams with *Q*=75.08. The good performance of the unigrams as feature can be explained with the lower number of OOV unigrams, compared to bigrams and trigrams. It is reasonable to expect that with large datasets the combined use of all three features will perform even better.

Table II 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 embedding is 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 doesn't require a language specific pre-trained word embedding and doesn't depend on the quality of such pre-trained embedding. The price for this is minimal hit in the performance, which can be increased with larger dataset. From this standpoint the third approach is the winner of the classifiers using embedding as features.

VI. DISCUSSION AND CONCLUSIONS

In this paper 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 $\alpha = 0.5$ was proposed. For DNN classifier using all n-grams as features when $\alpha = 1.0$ we have *WA*=94% and *UW*=42%. When the coefficient is moved to the other extreme, $\alpha = 0.0$, then weighted accuracy goes down to 67% and unweighted accuracy goes up to 69%. Using $\alpha = 0.5$ provides a good tradeoff between the WA and UA, where we have most of the gain in UA (up to 64%), without losing much from WA (down to 87%).

TABLE 1. RESULTS FOR BAG-OF-WORDS AS FEATURES

Classifier	Features	Validation set				Notes	Test set		
		WA	UA	TrTme	Q		WA	UA	WA+UA
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-,3-grams	88.15	58.78	73.3	73.34	4000 hu	89.37	59.20	74.29
	TF*IDF	89.68	56.61	27.5	73.12	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	TrT	Q		WA	UA	Q
Pre-trained on 16 Bill words	stats: #words, mean, min, max, stdev	ELM	88.7	56.23	169	72.28	5265 hu	89.7	59.1	74.37
		DNN	84.5	63.89	530	73.66	3 hl, 600 hu	85.8	65.9	75.85
	embeddings as a sequence	**LSTM + FC, last**	**88.2**	**62.18**	**233**	**74.94**	**150 hu**	**89.2**	**63.5**	76.32
		LSTM+sts+ELM	88.8	61.18	291	74.7	480 hu	89.8	63.1	76.42
Pre-trained on the dataset	train set	LSTM + FC, last	89.3	57.59	752	72.68	256hu	89.9	57.2	73.56
	tran+val sets	LSTM + FC, last	89.2	58.04	747	72.89	256hu	89.9	58.6	74.23
Embedding	trained jointly	**LSTM + FC, last**	**89.6**	**59.77**	**265**	**74.44**	**180 hu**	**90.3**	**58.4**	74.35

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.

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.

VI. ACKNOWLEDGEMENT

Authors would like to thank our colleagues Ashley Chang, Bryan Li, Dimitrios Dimitriadis, and Andreas Stolcke for labeling the data and fruitful discussions on 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*, vol. 37, pp. 98–125, 2017.

[2] B. Schuller, A. Batliner, S. Steidl, and D. Seppi, "Recognizing realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge," *Speech Communication*, vol. 53, no. 9, pp. 1062–1087, 2011.

[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] G. Mishne and 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.

[5] L. Oneto, F. Bisio, E. Cambria, and D. Anguita, "Statistical learning theory and ELM for big social data analysis," *IEEE Comput. Intell. Mag.*, vol. 11, no. 3, pp. 45–55, 2016.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013.

[7] 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.

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

[9] 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*, vol. 18, pp. 467–479, 1992.

[10] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[11] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.