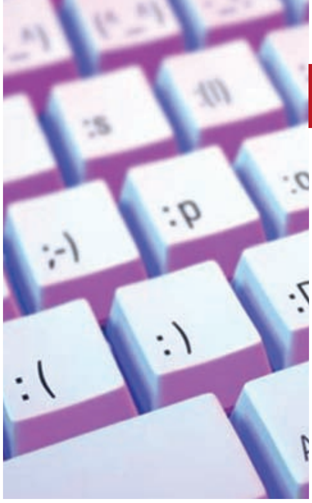

Editor: **Erik Cambria,** Nanyang Technological University, Singapore, cambria@ntu.edu.sg

Deep Learning-Based Document Modeling for Personality Detection from Text

Navonil Majumder, *Instituto Politécnico Nacional*
Soujanya Poria, *Nanyang Technological University*
Alexander Gelbukh, *Instituto Politécnico Nacional*
Erik Cambria, *Nanyang Technological University*

Personality is a combination of an individual's behavior, emotion, motivation, and thought-pattern characteristics. Our personality has great impact on our lives; it affects our life choices, well-being, health, and numerous other preferences. Automatic detection of a person's personality traits has many important practical applications. In the context of sentiment analysis,[1] for example, the products and services recommended to a person should be those that have been positively evaluated by other users with a similar personality type. Personality detection can also be exploited for word polarity disambiguation in sentiment lexicons,[2] as the same concept can convey different polarity to different types of people. In mental health diagnosis, certain diagnoses correlate with certain personality traits. In forensics, knowing personality traits helps reduce the circle of suspects. In human resources management, personality traits affect one's suitability for certain jobs.

Personality is typically formally described in terms of the Big Five personality traits,[3] which are the following binary (yes/no) values:

- *Extroversion* (EXT). Is the person outgoing, talkative, and energetic versus reserved and solitary?
- *Neuroticism* (NEU). Is the person sensitive and nervous versus secure and confident?
- *Agreeableness* (AGR). Is the person trustworthy, straightforward, generous, and modest versus unreliable, complicated, meager, and boastful?
- *Conscientiousness* (CON). Is the person efficient and organized versus sloppy and careless?
- *Openness* (OPN). Is the person inventive and curious versus dogmatic and cautious?

Texts often reflect various aspects of the author's personality. In this article, we present a method to extract personality traits from stream-of-consciousness essays using a convolutional neural network (CNN). We trained five different networks, all with the same architecture, for the five personality traits (see the "Previous Work in Personality Detection" sidebar for more information). Each network was a binary classifier that predicted the corresponding trait to be positive or negative.

To this end, we developed a novel document-modeling technique based on a CNN features extractor. Namely, we fed sentences from the essays to convolution filters to obtain the sentence model in the form of *n*-gram feature vectors. We represented each individual essay by aggregating the vectors of its sentences. We concatenated the obtained vectors with the Mairesse features,[4] which were extracted from the texts directly at the preprocessing stage; this improved the method's performance. Discarding emotionally neutral input sentences from the essays further improved the results.

For final classification, we fed this document vector into a fully connected neural network with one hidden layer. Our results outperformed the current state of the art for all five traits. Our implementation is publicly available and can be downloaded freely for research purposes (see http://github.com/senticnet/personality-detection).

1541-1672/17/$33.00 © 2017 IEEE
Published by the IEEE Computer Society

Previous Work in Personality Detection

The Big Five, also known as the Five Factor Model, is the most widely accepted model of personality. Initially, it was developed by several independent groups of researchers. However, it was advanced by Ernest Tupes and Raymond Christal[1]; J.M. Digman made further advancements,[2] and Lewis Goldberg later perfected it.[3]

Some earlier work on automated personality detection from plain text was done by James Pennebaker and Laura King,[4] who compiled the essay dataset that we used in our experiments (see http://web.archive.org/web/20160519045708/http://mypersonality.org/wiki/doku.php?id=wcpr13). For this, they collected stream-of-consciousness essays written by volunteers in a controlled environment and then asked the authors of the essays to define their own Big Five personality traits. They used Linguistic Inquiry and Word Count (LIWC) features to determine correlation between the essay and personality.[5]

François Mairesse and colleagues used, in addition to LIWC, other features, such as imageability, to improve performance.[6] Saif Mohammad and Svetlana Kiritchenko performed a thorough study on this essays dataset, as well as the MyPersonality Facebook status dataset, by applying different combinations of feature sets to outperform Mairesse's results, which they called the Mairesse baseline.[7]

Recently, Fei Liu and colleagues developed a language-independent and compositional model for personality trait recognition for short tweets.[8]

On the other hand, researchers have successfully used deep convolutional networks for related tasks such as sentiment analysis,[9] aspect extraction,[10] and multimodal emotion recognition.[11]

References

1. E. Tupes and R. Christal, *Recurrent Personality Factors Based on Trait Ratings*, tech. report ASD-TR-61-97, Lackland Air Force Base, 1961.
2. J. Digman, "Personality Structure: Emergence of the Five-Factor Model," *Ann. Rev. Psychology*, vol. 41, no. 1, 1990, pp. 417–440.
3. L. Goldberg, "The Structure of Phenotypic Personality Traits," *Am. Psychologist*, vol. 48, no. 1, 1993, pp. 26–34.
4. J.W. Pennebaker and L.A. King, "Linguistic Styles: Language Use as an Individual Difference," *J. Personality and Social Psychology*, vol. 77, no. 6, 1999, pp. 1296–1312.
5. J.W. Pennebaker, R.J. Booth, and M.E. Francis, *Linguistic Inquiry and Word Count: LIWC2007*, operator's manual, 2007.
6. F. Mairesse et al., "Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text," *J. Artificial Intelligence Research*, vol. 30, 2007, pp. 457–500.
7. S.M. Mohammad and S. Kiritchenko, "Using Hashtags to Capture Fine Emotion Categories from Tweets," *Computational Intelligence*, vol. 31, no. 2, 2015, pp. 301–326.
8. F. Liu, J. Perez, and S. Nowson, "A Language-Independent and Compositional Model for Personality Trait Recognition from Short Texts," *Computing Research Repository* (CoRR), 2016; http://arxiv.org/abs/1610.04345.
9. S. Poria et al., "A Deeper Look into Sarcastic Tweets using Deep Convolutional Neural Networks," *Proc. 26th Int'l Conf. Computational Linguistics*, 2016, pp. 1601–1612.
10. S. Poria, E. Cambria, and A. Gelbukh, "Aspect Extraction for Opinion Mining with a Deep Convolutional Neural Network," *Knowledge-Based Systems*, vol. 108, 2016, pp. 42–49.
11. S. Poria et al., "Convolutional MKL Based Multimodal Emotion Recognition and Sentiment Analysis," *Proc. IEEE Int'l Conf. Data Mining*, 2016, pp. 439–448.

Overview of the Method

Our method includes input data preprocessing and filtering, feature extraction, and classification. We use two types of features: a fixed number of document-level stylistic features, and per-word semantic features that are combined into a variable-length representation of the input text. This variable-length representation is fed into a CNN, where it is processed in a hierarchical manner by combining words into *n*-grams, *n*-grams into sentences, and sentences into a whole document. The obtained values are then combined with the document-level stylistic features to form the document representation used for final classification.

Specifically, our method includes the following steps:

- *Preprocessing*. This includes sentence splitting as well as data cleaning and unification, such as reduction to lowercase.
- *Document-level feature extraction*. We used the Mairesse baseline feature set, which includes such global features as the word count and average sentence length.
- *Filtering*. Some sentences in an essay may not carry any personality clues. Such sentences can be ignored in semantic feature extraction for two reasons: first, they represent noise that reduces the classifier's performance, and second, removal of those sentences considerably reduces the input size, and thus the training time, without negatively affecting the results. So, we remove such sentences before the next step.
- *Word-level feature extraction*. We represent individual words by word embedding in a continuous vector space; specifically, we experimented with the `word2vec` embeddings.[5] This gives a variable-length feature set for the document: the document is represented as a variable number of sentences, which are represented as a variable number of fixed-length word feature vectors.
- *Classification*. For classification, we use a deep CNN. Its initial layers process the text in a hierarchical manner. Each word is represented in the input as a fixed-length feature vector using `word2vec`, and sentences are represented as a variable number of word vectors. At some layer, this variable-length vector is reduced to fixed-length vector of each sentence, which is a kind of sentence embedding in a continuous vector space. At that level, documents are represented as a variable number of such fixed-length sentence embeddings. Finally, at a deeper layer, this variable-length document

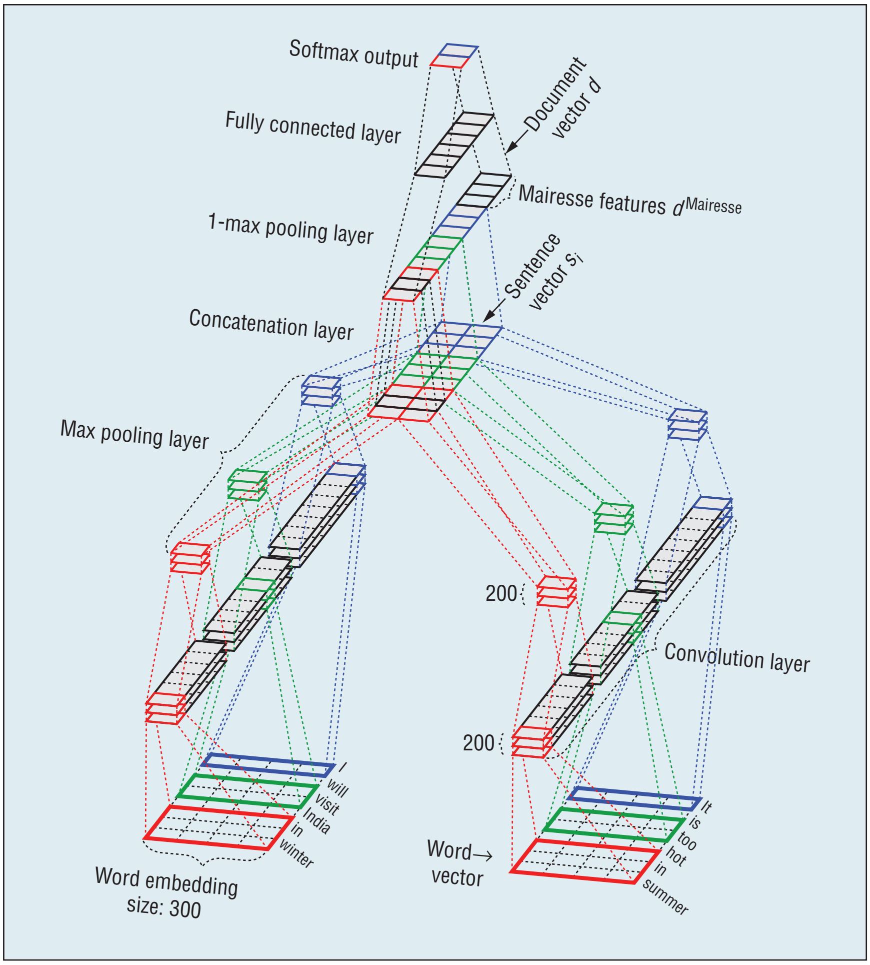

Figure 1. Architecture of our network. The network consists of seven layers. The input layer (shown at the bottom) corresponds to the sequence of input sentences (only two are shown). The next two layers include three parts, corresponding to trigrams, bigrams, and unigrams. The dotted lines delimit the area in a previous layer to which a neuron of the next layer is connected—for example, the bottom-right rectangle shows the area comprising three word vectors connected with a trigram neuron.

vector is reduced to a fixed-length document vector. This fixed-length feature vector is then concatenated with the document-level features giving a fixed-length document vector, which is then used for final classification.

When aggregating word vectors into sentence vectors, we use convolution to form word *n*-gram features. However, when aggregating sentence vectors into the document vector, we do not use convolution to form sentence *n*-gram features. We tried this arrangement, but the network did not converge in 75 epochs, so we left this experiment to our future work.

Network Architecture

We trained five separate neural classifiers, all with the same architecture, for the Big Five personality traits. The processing flow in our network comprises four main steps:

- word vectorization, in which we use fixed-length `word2vec` word embeddings as input data;
- sentence vectorization, from sequences of words in each sentence to fixed-length sentence vectors;
- document vectorization, from the sequence of sentence vectors to the document vector; and
- classification, from the document vector to the classification result (yes/no).

Accordingly, the network comprises seven layers: input (word vectorization), convolution (sentence vectorization), max pooling (sentence vectorization), 1-max pooling (document vectorization), concatenation (document vectorization), linear with Sigmoid activation (classification), and two-neuron softmax output (classification).

Figure 1 depicts the end-to-end network for two sentences. In the rest of this article, we discuss these steps and layers in detail.

Input

We represent the dataset as a set of documents: each document d is a sequence of sentences, each sentence s_i is a sequence of words, and each word w_j is a real-valued vector of fixed length known as word embedding. In our experiments, we used Google's pretrained `word2vec` embeddings.[5]

Thus, our input layer is a four-dimensional real-valued array from $\mathbb{R}^{D \times S \times W \times E}$, in which D is the number of documents in the dataset, S is the maximum number of sentences in a document across all documents, W is the maximum number of words in a sentence across all documents, and E is the length of word embeddings.

In implementation, to force all documents to contain the same number of sentences, we padded shorter documents with dummy sentences. Similarly, we padded shorter sentences with dummy words.

Aggregating Word Vectors into Sentence Vectors

We use three convolutional filters to extract unigram, bigram, and trigram features from each sentence. After max pooling, the sentence vector is a concatenation of the feature vectors obtained from these three convolutional filters.

Convolution. To extract the *n*-gram features, we apply a convolutional filter of size $n \times E$ on each sentence $s \in \mathbb{R}^{W \times E}$. We use 200 n-gram feature maps for each $n = 1, 2, 3$. So, for each n, our convolutional filter applied on the matrix s is $F_n^{conv} \in \mathbb{R}^{200 \times n \times E}$. We add a bias $B_n^{conv} \in \mathbb{R}^{200}$ to the output of the filter, which gives, for a given sentence, three feature maps $FM_n \in \mathbb{R}^{200 \times (W-n+1) \times 1}$, $n = 1, 2, 3$. To introduce nonlinearity, we apply the Rectified Linear Unit (ReLU) function to the feature maps FM_n.

Max pooling. Next, we apply max pooling to each feature map FM_n to further down-sample it to a feature map $DFM_n \in \mathbb{R}^{200 \times 1 \times 1}$, which we flatten to obtain a feature-vector of size 200.

Convolution. Finally, we concatenate the vectors obtained for the three types of n-gram to obtain a vector $s \in \mathbb{R}^{600}$ representing the sentence. We apply convolution and max pooling to each sentence in the document. The network parameters are shared between all sentences of the document. In particular, although we pad all sentences to a common size with dummy words, we do not need to pad all documents to a common size with dummy sentences.

Aggregating Sentence Vectors into a Document Vector

After individual sentences are processed, the document vector is a variable-sized concatenation of all its sentence vectors.

We assume that the document has some feature if at least one of its sentences has this feature. Each sentence is represented as a 600-dimensional vector. To obtain the document vector, for each of these 600 features, we take the maximum across all the sentences of the document. This gives a 600-dimensional real-valued vector $d^{network} \in \mathbb{R}^{600}$ of the whole document.

Adding Document-Level Features to Document Vector

François Mairesse and colleagues developed a document-level feature set for personality detection, consisting of 84 features.[4] It comprises the Linguistic Inquiry and Word Count features[6]; Medical Research Council features[7]; utterance-type features; and prosodic features. Examples of the features included in this set are the word count and average number of words per sentence, as well as the total number of pronouns, past tense verbs, present tense verbs, future tense verbs, letters, phonemes, syllables, questions, and assertions in the document.

We then concatenated those 84 features, d^{Mairesse}, with the document vector d^{network}. This gave the final 684-dimensional document vector $d = (d^{\mathrm{network}}, d^{\mathrm{Mairesse}}) \in \mathbb{R}^{684}$. We also used the feature set d^{Mairesse} as a baseline in our evaluation.

Classification

For final classification, we use a two-layer perceptron consisting of a fully connected layer of size 200 and the final softmax layer of size two, representing the *yes* and *no* classes.

Fully connected layer. We multiply the document $d \in \mathbb{R}^{684}$ by a matrix $W^{fc} \in \mathbb{R}^{684 \times 200}$ and add a bias $B^{fc} \in \mathbb{R}^{200}$ to obtain the vector $d^{fc} \in \mathbb{R}^{200}$. Introducing nonlinearity with Sigmoid activation improved the results:

$$\mathrm{d}^{fc} = \sigma(\mathrm{d}\ \mathrm{W}^{fc} + \mathrm{B}^{fc}),$$

where

$$\sigma(x) = 1/(1 + \exp(-x)).$$

We also experimented with ReLU and *tanh* as activation functions, but they yielded lower results.

Softmax output. We use the softmax function to determine the probability of the document to belong to the classes *yes* and *no*. For this, we build a vector

$$(x_{yes}, x_{no}) = d^{fc}\ W^{sm} + B^{sm},$$

where $W^{sm} \in \mathbb{R}^{200 \times 2}$ and the bias $B^{sm} \in \mathbb{R}^2$, and we calculate the class probabilities as

$$P(i \mid \text{network parameters}) = \frac{\exp(x_i)}{\exp(x_{\text{yes}}) + \exp(x_{\text{no}})}$$

for $i \in \{\text{yes, no}\}$.

Training

We use Negative Log Likelihood as the objective function for training. We randomly initialize the network parameters $F_1^{conv}, F_2^{conv}, F_3^{conv}, B_1^{conv}, B_2^{conv}, B_3^{conv}$, W^{fc}, B^{fc}, W^{sm}, and B^{sm}. We use Stochastic Gradient Descent with Adadelta[8] update rules to tune the network parameters in order to minimize the error defined as negative log likelihood. In our experiments, after 50 epochs, the network converged, with 98 percent training accuracy.

Experimental Results

To evaluate our method, we tested it on a well-known dataset typically used to compare personality detection techniques.

Dataset

We used James Pennebaker and Laura King's stream-of-consciousness essay dataset.[6] It contains 2,468 anonymous essays tagged with the authors' personality traits: EXT, NEU, AGR, CON, and OPN. We removed from the dataset one essay that contained only the text "Err:508," and we experimented with the remaining 2,467 essays.

Experimental Setting

In all of our experiments, we used tenfold cross-validation to evaluate the trained network.

Table 1. Accuracy obtained with different configurations.

Document vector *d*	Filter	Classifier	Convolution filter	Personality traits				
				EXT	NEU	AGR	CON	OPN
N/A	N/A	Majority	N/A	51.72	50.02	53.10	50.79	51.52
Word *n*-grams	Not used	SVM	N/A	51.72	50.26	53.10	50.79	51.52
Mairesse[12]	N/A	SVM	N/A	55.13	58.09	55.35	55.28	59.57
Mairesse (our experiments)	N/A	SVM	N/A	55.82	58.74	55.70	55.25	60.40
Published state of the art per trait[12]	N/A	N/A	N/A	56.45	58.33	56.03	**56.73**	60.68
CNN	N/A	MLP	1, 2, 3	55.43	55.08	54.51	54.28	61.03
CNN	N/A	MLP	2, 3, 4	55.73	55.80	55.36	55.69	61.73
CNN	N/A	SVM	2, 3, 4	54.42	55.47	55.13	54.60	59.15
CNN + Mairesse	N/A	MLP	1, 2, 3	54.15	57.58	54.64	55.73	61.79
CNN + Mairesse	N/A	SVM	1, 2, 3	55.06	56.74	53.56	56.05	59.51
CNN + Mairesse	N/A	sMLP/FC	1, 2, 3	54.61	57.81	55.84	**57.30**	62.13
CNN + Mairesse	Used	sMLP/MP	1, 2, 3	**58.09**	57.33	**56.71**	56.71	61.13
CNN + Mairesse	Used	MLP	1, 2, 3	55.54	58.42	55.40	56.30	**62.68**
CNN + Mairesse	Used	SVM	1, 2, 3	55.65	55.57	52.40	55.05	58.92
CNN + Mairesse	Used	MLP	2, 3, 4	55.07	**59.38**	55.08	55.14	60.51
CNN + Mairesse	Used	SVM	2, 3, 4	56.41	55.61	54.79	55.69	61.52
CNN + Mairesse	Used	MLP	3, 4, 5	55.38	58.04	55.39	56.49	61.14
CNN + Mairesse	Used	SVM	3, 4, 5	56.06	55.96	54.16	55.47	60.67

*Bold indicates the best result for each trait.

Preprocessing. We split the text into a sequence of sentences at the period and question mark characters. Then we split each sentence into words at whitespace characters. We reduced all letters to lowercase and removed all characters other than ASCII letters, digits, exclamation marks, and single and double quotation marks.

Some essays in the dataset contained no periods or missing periods, resulting in absurdly long sentences. For these cases, we split each obtained "sentence" that was longer than 150 words into "sentences" of 20 words each (except the last piece that could happen to be shorter).

Extracting document-level features. We used Mairesse and colleagues' library (http://farm2.user.srcf.net/research/personality/recognizer.html) to extract the 84 Mairesse features from each document.[4]

Sentence filtering. We assumed that a relevant sentence would have at least one emotionally charged word. After extracting the document-level features, but before extracting the `word2vec` features, we discarded all sentences that had no emotionally charged words.

We used the NRC Emotion Lexicon (http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm) to obtain emotionally charged words.[9,10] This lexicon contains 14,182 words tagged with 10 attributes: anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise, and trust. We considered a word to be emotionally charged if it had at least one of these attributes; there are 6,468 such words in the lexicon (most of the words in this lexicon have no attributes).

So, if a sentence contained none of the 6,468 words, we removed it before extracting the `word2vec` features from the text. In our dataset, all essays contained at least one emotionally charged word.

We also experimented with not removing any sentences and with randomly removing half of each essay's sentences. Randomly removing half of the sentences improved the results as compared with no filtering at all; we do not have a plausible explanation for this fact. Removing emotionally neutral sentences as described earlier further improved the results, producing the best results for all five traits. Filtering also improved the training time by 33.3 percent.

Extracting word-level features. We used the `word2vec` embeddings[5] (http://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit) to convert words into 300-dimensional vectors. If a word was not found in the list, we assigned all 300 coordinates randomly with a uniform distribution in [−0.25, 0.25].

Word *n*-gram baseline. As a baseline feature set, we used 30,000 features: 10,000 most-frequent-word unigrams, bigrams, and trigrams in our dataset. We used the Scikit-learn library to extract these features from the documents.[11]

Classification. We experimented with three classification settings. In the variant marked MLP in Table 1, we used the network shown in Figure 1, which is a multiple-layer perceptron (MLP) with one hidden layer, trained together with the CNN. In the variant marked SVM (support vector machine) in the table, we first trained the network shown in Figure 1 to obtain the corresponding document vector *d* for each document in the dataset, and then used these vectors to train a polynomial SVM of degree 3. In the variant marked sMLP/MP in the table, in a similar manner we used the vectors *d* (the max pooling layer) to train a stand-alone MLP (using 50 epochs) with the same configuration as the last two layers in Figure 1 (that is, using the 1-max pool layer from Figure 1 as input).

In another experiment, we fed to the stand-alone MLP the values from the fully connected layer instead of d; this variant is marked as sMLP/FC in Table 1. For baseline experiments not involving the use of CNN, we used only a linear SVM.

Results

Table 1 shows our results. Our method outperformed the state of the art for all five traits, although with different configurations for different traits.

Using n-grams showed no improvement over the majority baseline: the classifier rejected all n-grams. Applying filtering and adding the document-level (Mairesse) features proved to be beneficial. In fact, the CNN alone without the document-level features underperformed the Mairesse baseline. We attribute this to insufficient training data: our training corpus was only 1.9 million running words.

Contrary to our expectations, applying SVM to the document vector d built with the CNN did not improve the results. Surprisingly, applying a stand-alone MLP to d improved the results. We cannot attribute this to the fact that the system had thus received an additional 50 epochs of training, because the network used to build the document vector d has converged in its 50 epochs of initial training.

Increasing the window size for convolution filters did not seem to consistently improve the results; while the best result for the NEU trait was obtained with 2-, 3-, and 4-grams, even sizes 1, 2, and 3 outperformed the current state of the art.

We also tried several configurations not shown in Table 1, as well as some variations of the network architecture. In particular, in addition to using convolution filters to obtain a vector for each sentence, we tried using convolution filters to obtain document vector d from the sequence of sentence vectors s_i. However, training did not converge after 75 epochs, so we used 1-max pooling layer on the array of sentence vectors to obtain the document vector.

In the future, we plan to incorporate more features and preprocessing. We plan to apply the Long Short Term Memory (LSTM) recurrent network to build both the sentence vector from a sequence of word vectors and the document vector from a sequence of sentence vectors. In addition, we plan to apply our document modeling technique to other emotion-related tasks, such as sentiment analysis or mood classification.[13] ■

References

1. E. Cambria, "Affective Computing and Sentiment Analysis," *IEEE Intelligent Systems*, vol. 31, no. 2, 2016, pp. 102–107.
2. E. Cambria et al., "SenticNet 4: A Semantic Resource for Sentiment Analysis based on Conceptual Primitives," *Proc. 26th Int'l Conf. Computational Linguistics*, 2016, pp. 2666–2677.
3. J. Digman, "Personality Structure: Emergence of the Five-Factor Model," *Ann. Rev. Psychology*, vol. 41, 1990, pp. 417–440.
4. F. Mairesse et al., "Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text," *J. Artificial Intelligence Research*, vol. 30, 2007, pp. 457–500.
5. T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," *Computing Research Repository* (CoRR), 2013; http://arxiv.org/abs/1301.3781.
6. J.W. Pennebaker and L.A. King, "Linguistic Styles: Language Use as an Individual Difference," *J. Personality and Social Psychology*, vol. 77, no. 6, 1999, pp. 1296–1312.
7. M. Coltheart, "The MRC Psycholinguistic Database," *Quarterly J. Experimental Psychology*, vol. 33A, 1981, pp. 497–505.
8. M.D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *Computing Research Repository* (CoRR), 2012; https://arxiv.org/abs/1212.5701.
9. S.M. Mohammad and P.D. Turney, "Crowdsourcing a Word-Emotion Association Lexicon," *Computational Intelligence*, vol. 29, no. 3, 2013, pp. 436–465.
10. S. Mohammad and P. Turney, "Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon," *Proc. NAACL-HLT Workshop Computational Approaches to Analysis and Generation of Emotion in Text*, 2010, pp. 26–34.
11. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *J. Machine Learning Research*, vol. 12, Oct. 2011, pp. 2825–2830.
12. S.M. Mohammad and S. Kiritchenko, "Using Hashtags to Capture Fine Emotion Categories from Tweets," *Computational Intelligence*, vol. 31, no. 2, 2015, pp. 301–326.
13. B.G. Patra, D. Das, and S. Bandyopadhyay, "Multimodal Mood Classification Framework for Hindi Songs," *Computación y Sistemas*, vol. 20, no. 3, 2016, pp. 515–526.

Navonil Majumder is a postgraduate student at the Centro de Investigación en Computación (CIC) of the Instituto Politécnico Nacional, Mexico. Contact him at navo@nlp.cic.ipn.mx.

Soujanya Poria is a research assistant at Temasek Laboratories at Nanyang Technological University. Contact him at sporia@ntu.edu.sg.

Alexander Gelbukh is a research professor at the CIC of the Instituto Politécnico Nacional. Contact him at gelbukh@cic.ipn.mx.

Erik Cambria is an assistant professor in the School of Computer Science and Engineering at Nanyang Technological University. He is also affiliated with the Rolls-Royce@NTU Corporate Lab, A*STAR SIMTech, and MIT Synthetic Intelligence Lab. Contact him at cambria@ntu.edu.sg.