Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06542-1

ORIGINAL ARTICLE

Emotionally charged text classification with deep learning and sentiment semantic

Jeow Li Huan[1] · Arif Ahmed Sekh[2,3] · Chai Quek[1] · Dilip K. Prasad[2]

Received: 3 December 2020 / Accepted: 9 September 2021

Abstract
Text classification is one of the widely used phenomena in different natural language processing tasks. State-of-the-art text classifiers use the vector space model for extracting features. Recent progress in deep models, recurrent neural networks those preserve the positional relationship among words achieve a higher accuracy. To push text classification accuracy even higher, multi-dimensional document representation, such as vector sequences or matrices combined with document sentiment, should be explored. In this paper, we show that documents can be represented as a sequence of vectors carrying semantic meaning and classified using a recurrent neural network that recognizes long-range relationships. We show that in this representation, additional sentiment vectors can be easily attached as a fully connected layer to the word vectors to further improve classification accuracy. On the UCI sentiment labelled dataset, using the sequence of vectors alone achieved an accuracy of 85.6%, which is better than 80.7% from ridge regression classifier—the best among the classical technique we tested. Additional sentiment information further increases accuracy to 86.3%. On our suicide notes dataset, the best classical technique—the Naíve Bayes Bernoulli classifier, achieves accuracy of 71.3%, while our classifier, incorporating semantic and sentiment information, exceeds that at 75% accuracy.

Keywords Text classification · Sentiment analysis · LSTM

1 Introduction

Text classification is the task of organizing text documents into pre-defined categories [1]. It is an important aspect of data processing to make data usable by humans and is used in spam filtering [2], language identification [3], sentiment analysis [4] and many other areas. The Naíve Bayes classifier is a popular and effective algorithm for text classification [5–7]. Other general classifiers can also be adapted for text classification by using the vector space model [8]. Popular general classifiers include support vector classifier [9] and stochastic gradient descent classifier [10]. They work on the basis of finding a hyperplane that best separates data points from two classes. Their linear classifier variant works well on text documents and often performs better than Naíve Bayes. However, Naíve Bayes and vector space model discard the position of words and cannot capture the relationship between words. While n-words or n-grams can be used, they cannot capture long-range relationships [11]. Latent semantic indexing solves the problem through the application of singular value decomposition [12].

A recent survey articles [13, 14] suggest deep neural network is playing a vital role in recent language processing tasks. The methods are successfully applied in many tasks such as sentiment analysis, spam filtering and marketing. This also leads us to use deep leaning in our method. Recently, Jiang et al. [15] proposed a Focal Loss-based which is used in sentiment analysis. Chatterjee et al. [16] extend similar task into big data framework. Lu et al. use a bidirectional LSTM [17]. A semantic-based feature selection model is proposed in [2]. Associative rule-based systems are also achieved state-of-the-art accuracy in sentiment classification [7]. It is noted that new way of distance measurement can improve the classification models [8].

✉ Arif Ahmed Sekh
skarifahmed@gmail.com

[1] Nanyang Technological University, Nanyang Avenue, Singapore

[2] UiT The Arctic University of Norway, Tromsø, Norway

[3] XIM University, Bhubaneswar, Odisha, India

Published online: 28 September 2021

State-of-the-art techniques treat text as one-dimensional, either as word–probability pairs in Naíve Bayes or as a document vector (Fig. 1, existing method). To push text classification accuracy even higher, two-dimensional document representation, such as vector sequences or matrices, should be explored (Fig. 1, proposed method). Many text classification algorithms operate on document vectors. The dimension of the vector needs to be kept low for better generalization. Converting a document to fit a single low-dimensional document vector necessitates discarding much data. Positions, plurality and tenses of words are often omitted.

We seek to overcome these limitations by representing a document as a sequence of vectors instead. The state-of-the-art recurrent neural network—long short-term memory (LSTM) [18], will be used to build the classifier. Using a sequence of vectors provides the flexibility to preserve as much information from the original text as possible, and to incorporate information from external databases to supplement the text. We incorporate semantic information from pre-trained GloVe vectors [19], and sentiment information from SentiWordNet [20] to mirror the way human comprehends text. A person goes through formal education to learn the meaning of words. When he/she reads a document, these learned meanings enable comprehension.

The overall aim is to reach a classification accuracy beyond what the classic approaches are able to achieve. Classic approaches include Naíve Bayes, support vector, stochastic gradient descent, passive–aggressive, k-nearest neighbour, Rocchio and ridge regression classifiers. We experimentally proved that attachment of sentiment vector (emotionally charged) and the accuracy of state-of-the-art text classifiers perform better.

1.1 Background

The Naíve Bayes classifiers use the Bayes rule in computing the probability that a document belongs to a class. They assume that occurrence of each word is independent of other words. They compute the probability $P(C_k|w_1,\ldots,w_n) \propto P(C_k)\prod_{i=1}^{n} P(w_i|C_k)$ for each class C, where w_i is the i^{th} word in the dictionary. Since multiplication is commutative, word positions do not matter. A document can be seen as a set of words, thus one-dimensional. Support vector, stochastic gradient descent (SGD) and passive–aggressive classifiers find the best separating hyperplanes. A document thus has to be represented as a point in multi-dimensional space (i.e. a vector) using these steps:

1. The first step is to tokenize the document. A document can be seen in its entirety, or as its sections, paragraphs, sentences, words or characters. Since words are the smallest unit that have meaning, most classification works on the word level. For English documents, one can extract words by breaking the document at every occurrence of space, comma, period and other word boundaries. Some tokenizers use more complex algorithms to preserve hyphenated words, such as "ice-cream", abbreviations, such as "don't" and "U.S.", and non-words, such as "1/2", "john@example.com" and "http://example.com". Some tokenizers preserve punctuations as well. English tokenizers include the *StandardAnalyzer* in the Lucene library [21] and the *PTBTokenizer* in the Stanford Natural Language Processing library [22].
2. Stemming is then performed to turn the words into their root forms, so as to discard the plurality and tense. Stemming helps reduce the dimension of the document vector. Stemming may use heuristics, such as in Porter Stemmer [23], or morphology, such as in the *Morphy* class in Stanford Natural Language Processing library [24].
3. A dictionary is constructed by listing the unique words in all the documents. A document is then represented by a vector of dimension equal to the size of the

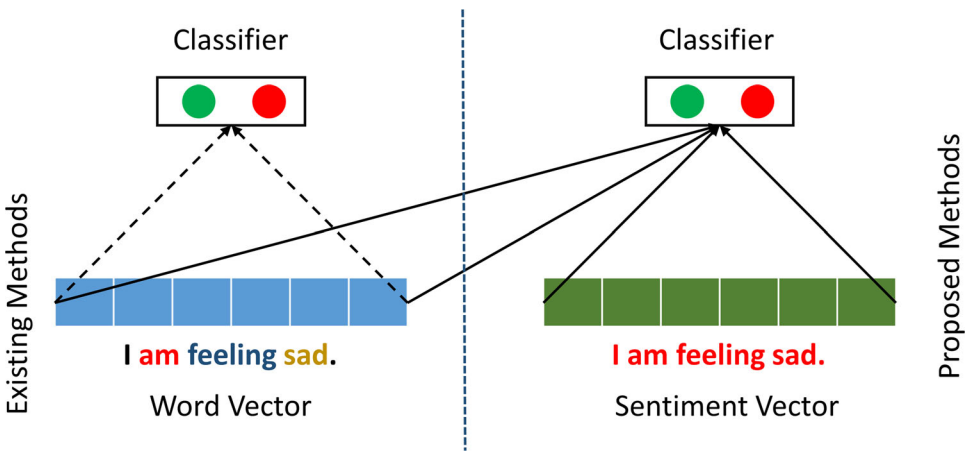

Fig. 1 Demonstration of the proposed solution. We use sentiment vector addition to the word vector

dictionary. The n^{th} element in the vector corresponds to the n^{th} word in the dictionary. In the bag-of-word (BoW) model, each vector element stores term frequency, which is the number of occurrence of the corresponding word in the document. However, it assigns large numbers for these frequent words, such as "a" and "the", that do not carry much information, skewing distance calculation between vectors. The problem can be solved by dividing the term frequency by the document frequency, where the document frequency is the number of documents containing the word. This yields the term frequency—inverse document frequency (tf·idf) statistic. Using these vector space models (BoW and tf·idf) discards the position of the words in the document.

The k-nearest neighbour classifier also uses the vector space model. It classifies a given point as the most frequently occurring class among k-closest neighbours [25]. Rocchio classifier is the nearest centroid classifier with special vector to represent documents. For each class, the centroids (mean) of the labelled data points are computed. A data point is then classified as the class with the nearest centroid to the point [26]. Ridge regression classifier uses ridge regression as a classifier. In standard regression, the error function is $J_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y - \mathbf{w}^T\mathbf{x}_i)^2$. Ridge regressions add a regularization factor to penalize nonzero weights proportional to a shrinkage coefficient λ, giving $J_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda\|\mathbf{w}\|^2$. The proposed classification is demonstrated in Fig. 1.

2 Proposed method

Here, we propose a text classifier that uses a dual modality of information extraction and a long short-term memory recurrent neural network (LSTM) for the classification. Firstly, a word embedding feature is extracted from pre-trained model. Next, the emotion of text is extracted from sentiment network. Finally, the features are combined to classify the text. An LSTM is a type of artificial neural network with self-connection and nodes made up of gated memory blocks. The proposed method is depicted in Fig. 2.

2.1 LSTM-based classifier with sentiment data

A long short-term memory neural network (LSTM) is an RNN with memory cells and gates units [27]. The recurrent network used in the proposed model is presented in Fig. 3. A node in the hidden layer is in itself a network that consists of input gate, output gate, forget gate, a memory cell and a self-recurrent connection on the memory cell (Fig. 4).

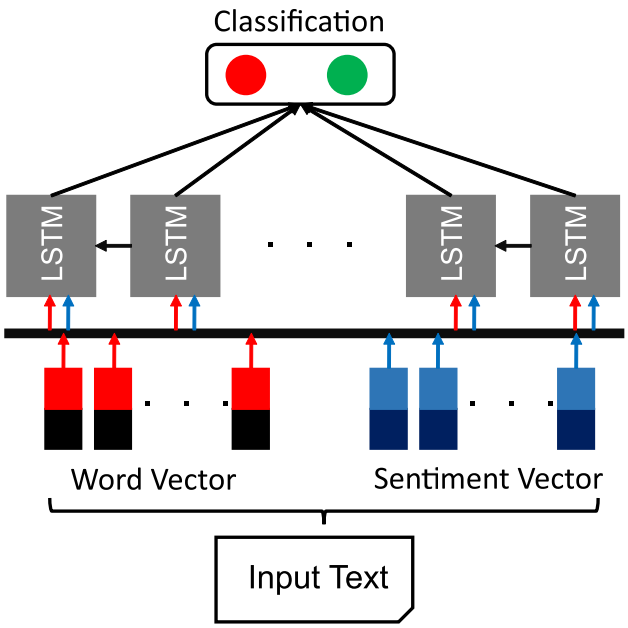

Fig. 2 Proposed LSTM-based classifier

LSTM solves the exploding gradient problem by truncating the gradient computation. It solves vanishing gradient by having the memory cell to repeat its state across time, and turning on and off the input and output gates to allow or prevent modifications to the memory cell. Several variants of LSTM exist; some has additional features such as peephole connections, while other has less, such as removal of output activation function. Klaus Greff et al. found that removing peephole connection and full gate recurrence simplifies computation without affecting performance [28].

Overfitting in LSTM is usually overcome by using dropout. Wojciech Zaremba et al. found that in RNN—including LSTM—dropout works best when applied only on non-recurrent connections [29].

The values in the node are computed as follows:

$$\begin{aligned}
\text{Block input}: &\quad z^t = g(\mathbf{W}_z x^t + \mathbf{R}_z y^{t-1} + b_z)\\
\text{Input gate}: &\quad i^t = \sigma(\mathbf{W}_i x^t + \mathbf{R}_i y^{t-1} + b_i)\\
\text{Forget gate}: &\quad f^t = \sigma(\mathbf{W}_f x^t + \mathbf{R}_f y^{t-1} + b_f)\\
\text{Cell state}: &\quad c^t = i^t \odot z^t + f^t \odot c^{t-1}\\
\text{Output gate}: &\quad o^t = \sigma(\mathbf{W}_o x^t + \mathbf{R}_o y^{t-1} + b_o)\\
\text{Block output}: &\quad y^t = o^t \odot h(c^t)
\end{aligned}$$

where $\mathbf{W}$ are rectangular input weight matrices, $\mathbf{R}$ are square recurrent weight matrices, and b are bias vectors. Functions σ, g and h are pointwise nonlinear activation function, and $\odot$ is pointwise multiplication of two vectors. The sigmoid function σ is defined as $\sigma(t) = \frac{1}{1+e^{-t}}$. The hyperbolic tangent (tanh) function is defined as

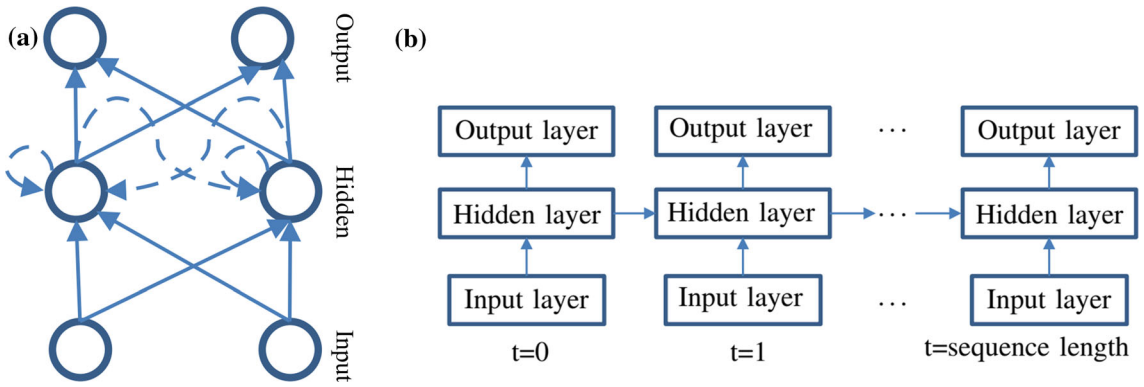

Fig. 3 **a** A fully connected RNN used as the foundation of the proposed method. **b** The proposed model uses RNN as an infinitely deep feedforward neural network

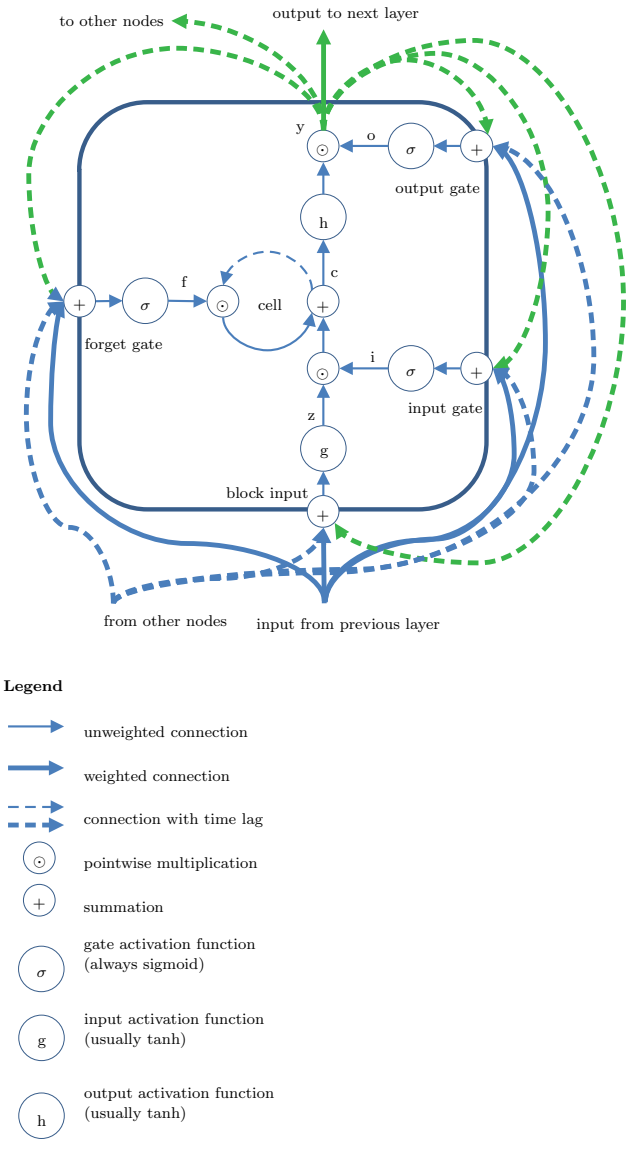

Fig. 4 The network inside a long short-term memory block

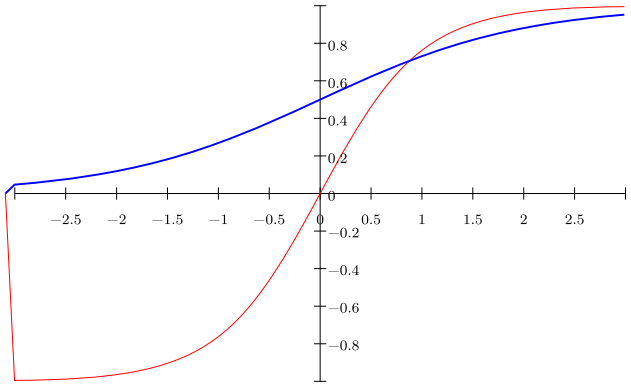

Fig. 5 Graphs of sigmoid (thick, blue) and hyperbolic tangent (thin, red) functions

$\tanh(t) = \frac{1-e^{-2t}}{1+e^{-2t}}$. The activation function is demonstrated in Fig. 5.

Sequence and non-sequence data have to be processed by different types of nodes. To process sequence data consisting of GloVe vectors and/or SentiWordNet scores, 1 hidden layer of 120 LSTM nodes is used (Figs. 6 and 7). The memory block is illustrated in Fig. 4. Gradients above 100 are clipped to prevent exploding gradient. To prevent overfitting, input layer is first fed into a dropout layer that has 50% probability of setting value to zero and rescales input with $output = \frac{input}{1-0.5}$. Dropout layer is turned off during cross-validation by setting probability of dropout to zero. To process non-sequence valence–arousal data, 1 hidden layer of 3 nodes with tanh activation is used. The output from both the LSTM and tanh nodes is then concatenated via the concat layer, before connecting to the output layer. Learning rate is initialized at 0.01, and algorithm is set as AdaGrad. The Lasagne library is used to create the architecture [30]. It abstracts the layers and encapsulates the mathematical computations of the neural network. We have used Lasagne, which is in turn built on

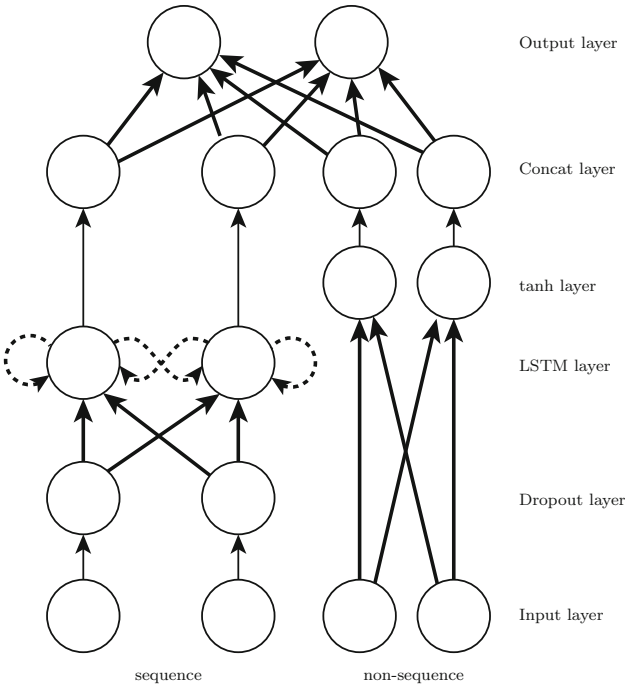

Fig. 6 LSTM architecture used in the experiment. As a simplification, only two nodes per layer are illustrated

Theano, which is a library that uses GPU or CPU to accelerate computations [31].

Semantic information: Each document is tokenized using the `PTBTokenizer` in the Stanford Natural Language Processing library. The GloVe vectors and SentiWordNet scores are then retrieved for the tokens. To compare the effect of having semantic and sentiment information against not having them, we compare against one-hot encoding of the tokens. GloVe performs unsupervised learning on documents to obtain word–word co-occurrence statistics and represent it in a vector [19]. This results in words similar in meaning represented with vectors that are of small distance to each other. Co-occurrence probabilities can be modelled in the general form using equation (1).

$$F(w_i, w_j, \tilde{w_k}) = \frac{P_i k}{P_j k} \tag{1}$$

where $w \in R^d$ are the word vectors,

$\tilde{w} \in R^d$ are separate context word vectors, and

$P_{ij} = P(i|j) = \frac{x_{ij}}{x_i}$ is the probability that word j appears in the context of word i.

Computation of word vector is by minimizing the weighted least squares regression model:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b_j} - \log X_{ij})^2 \tag{2}$$

where V is the size of the vocabulary, f is the weighting function, b is the bias for w, and $\tilde{b}$ is the bias for $\tilde{w}$. As a neural network is used to solve this model, different random initializations would yield different results. To achieve consistent performance, separate context word vectors $\tilde{w}$ are trained on the same neural network but with different random initializations.

To measure distance in text classification, cosine distance is often used. Cosine distance is unaffected by the magnitude of the vector, unlike Euclidean distance. Consider the case in the bag-of-word and tf·idf models, magnitude can be doubled by appending a duplicate of a document to itself (Fig. 8). The document with double the magnitude has the exact same content, albeit duplicated, as the original document, and thus should be regarded similar.

The cosine distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$d = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2} \tag{3}$$

To illustrate that words with similar meaning having closer vectors, we compute the cosine distance between "water", "ice", "learn" and "educate" (Table 1). The distance between "water" and "ice" and between "learn" and "educate" is lower than other permutation.

We used pre-trained 300-dimensional GloVe vectors trained on Wikipedia 2014 and Gigaword 5 [32]. We also tested the 50-dimensional vectors, but found them ineffective. This set of pre-trained vectors contains 6 billion tokens, of which 400 thousand is vocabulary in lower case. The rest of the tokens are punctuations, numbers, dates, email addresses and others. It is 989 MiB uncompressed. A dictionary is constructed by listing the unique words in all the documents. A word is then represented by a vector of dimension equal to the size of the dictionary. A word that is at n^{th} position in the dictionary is represented by a vector with one at the n^{th} element and zero at all other elements.

Sentiment information: We are interested in classifying emotionally charged documents. Intuitively, additional sentiment information should help classification. We used SentiWordNet to provide positivity, negativity and objectivity scores. For our smaller dataset, we also had 12 healthy participants rate the amount of valence and arousal (approximating the circumplex model of affect) of the documents. WordNet is a database that groups words into sets of synonyms [33]. Similarity between words can be inferred from its link to the synonym sets and the number of linkages between the synonym sets. SentiWordNet is built on top of WordNet and its similarity relationship [34]. It adds positivity, negativity and objective score to each synonym set. Subjectivity of a word can thus be found by identifying the synonym set it belongs to and looking up

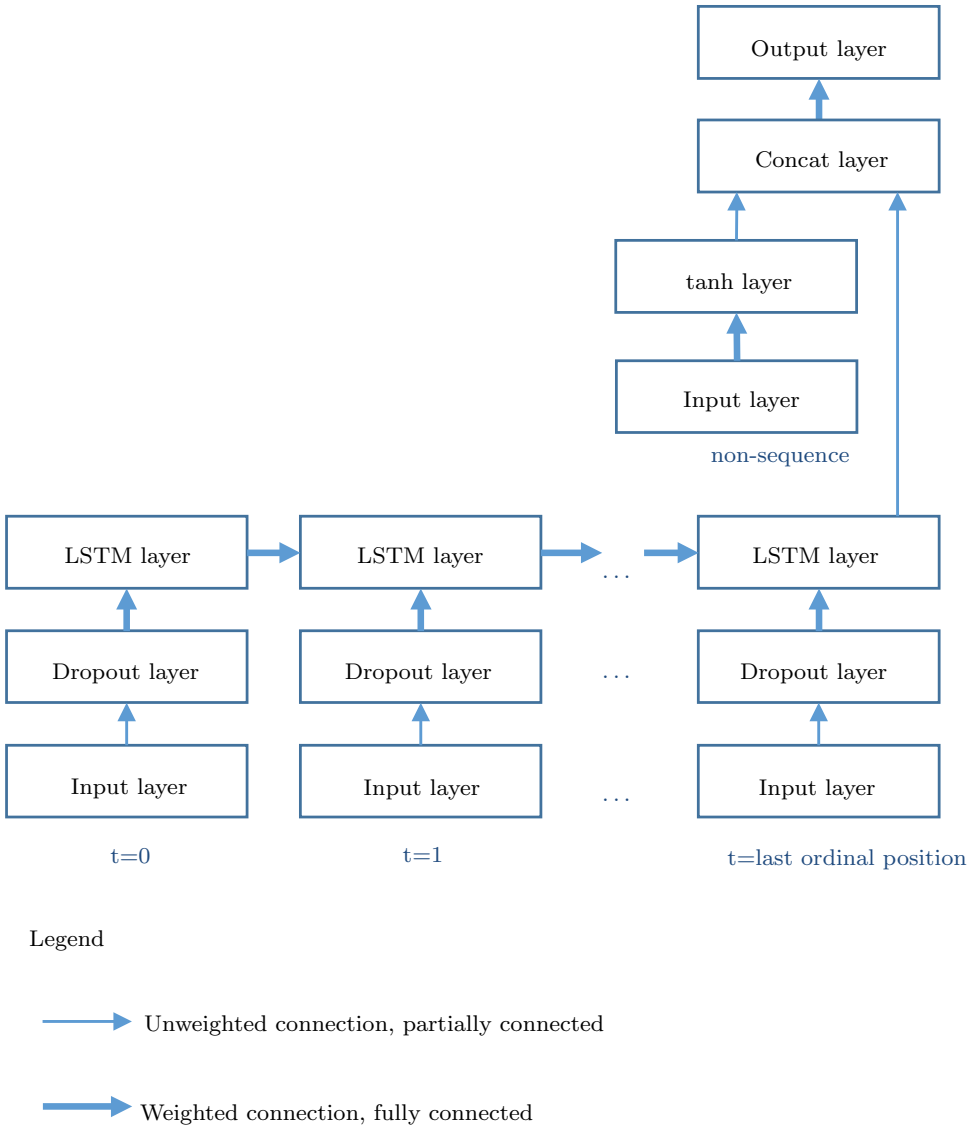

Fig. 7 LSTM architecture used in the experiment, illustrated with the recurrent connections unrolled into an indefinitely deep feedforward network

the three scores. As an example, the synonym sets for "good" are:

good.n.01	good.n.02	good.n.03
commodity.n.01	good.a.01	full.s.06
good.a.03	estimable.s.02	beneficial.s.01
good.s.06	good.s.07	adept.s.01
good.s.09	dear.s.02	dependable.s.04
good.s.12	good.s.13	effective.s.04
good.s.15	good.s.16	good.s.17
good.s.18	good.s.19	good.s.20
good.s.21	well.r.01	thoroughly.r.02

The list contains several meanings, such as goodness and product, and parts of speech, such as noun (n), adjective (a), adjective satellite (s) and adverb (r). Of these synonym sets, SentiWordNet labels a subset of them with sentiment scores.

$$
\begin{aligned}
&<\text{good.n.01: PosScore} = 0.5 \text{NegScore} = 0.0> \\
&<\text{good.n.02: PosScore} = 0.875 \text{NegScore} = 0.0> \\
&<\text{good.n.03: PosScore} = 0.625 \text{NegScore} = 0.0> \\
&<\text{commodity.n.01: PosScore} = 0.0 \text{NegScore} = 0.0> \\
&<\text{good.a.01: PosScore} = 0.75 \text{NegScore} = 0.0> \\
&<\text{good.a.03: PosScore} = 1.0 \text{NegScore} = 0.0> \\
&<\text{well.r.01: PosScore} = 0.375 \text{NegScore} = 0.0> \\
&<\text{thoroughly.r.02: PosScore} = 0.0 \text{NegScore} = 0.0>
\end{aligned}
$$

Objective score is computed from $1 - PosScore - NegScore$. We used SentiWordNet 3. It is based on WordNet 3, which has 117 thousand synonym sets. SentiWordNet 3 is build using 1105 synonym sets that

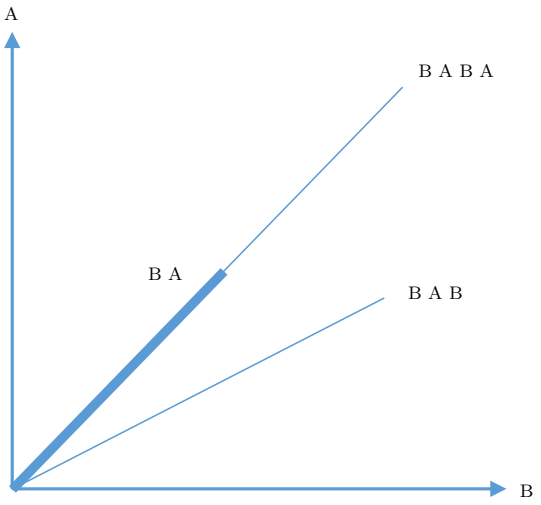

Fig. 8 Euclidean distance would consider the document containing the words B A B A as closer to the document B A B than B A. However, intuitively, appending a document to itself does not fundamentally change it, and thus, B A B A should be more similar to B A than B A B. Cosine distance (the angular difference between the vectors) conforms to this intuition

Table 1 Cosine distance between GloVe vectors of water, ice, learn and educate

	Water	Ice	Learn	Educate
Water	0	**0.597**	0.844	0.907
Ice	**0.597**	0	0.905	1.054
Learn	0.844	0.905	0	**0.493**
Educate	0.907	1.054	**0.493**	0

are carefully chosen and then scored manually. A random walk algorithm that visits the links between synonym sets propagates these scores to other synonym sets. We encountered an issue obtaining the SentiWordNet scores. A word can have many meanings and thus belong to multiple synonym sets in WordNet. Disambiguating among the different synonym sets is hard, even by hand. As a shortcut, we obtained positivity, negativity and objectivity scores by taking the average score in all synonym sets of a word.

Circumplex model of affect: The circumplex model of affect proposes that all affective states emerge from cognitive interpretations of neural sensations from two independent neurophysiological systems [35]. One system causes valence, while the other causes activation. It places emotions in a two-dimensional circular space, with one dimension indicating the level of valence and another the level of arousal (Fig. 9). Neutral point is at the centre of the circular space.

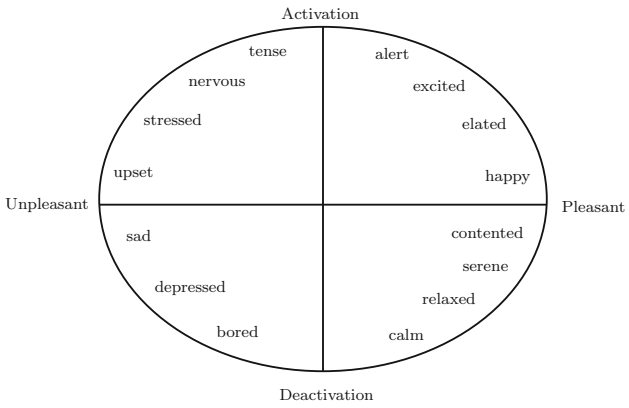

Fig. 9 Various emotions placed on the circumplex model of affect

To simplify survey design, we collected valence and arousal ratings separately, with each scale ranging from -5 to 5. This approximation avoids having to design a custom input that presents the circular scale of the model.

3 UCI sentiment labelled sentences

The University of California, Irvine Sentiment Labelled Sentences consist of positive and negative labelled sentences taken from three websites: imdb.com, amazon.com and yelp.com [36]. For each website, there are 500 positive sentences (labelled as 1) and 500 negative sentences (labelled as 0).

Parameters: Documents were tokenized into words using the `PTBTokenizer`. To obtain document vectors, the tokens were stemmed using Porter stemmer, and then, the tf·idf statistic was computed. To obtain one-hot vectors, the tokens were similarly stemmed. The parameters used for the classifiers were

K-nearest neighbour:	10 nearest neighbour
Passive aggressive:	trained for 50 iterations
SGD:	trained for 50 iterations
LSTM:	trained for 6 iterations

Spell correction for pre-trained GloVe and SentiWordNet: This dataset has not been pre-processed and contains several misspellings and missing spacing around punctuations. Missing spacing results in `PTBTokenizer` not splitting some text into separate tokens. This incorrect tokenization and misspelling causes failure to find the pre-trained GloVe vectors and the SentiWordNet scores. We used the Aspell 0.60.6.1 in `bad-spellers` mode and Algorithm 1 to correct the problems.

Algorithm 1 Procedure to retrieve GloVe vectors

```
list = tokenize(document)
while list is not empty do
  pop a token from list
  if token found in pre-trained GloVe then
    get GloVe vector
  else
    if token contains alpha-numeric characters then
      if token contains non-alpha-numeric characters then
        token = remove non-alpha-numeric characters
        if token found in pre-trained GloVe then
          get GloVe vector
        else
          parts = split token at word boundary
          push parts into list
        end if
      else
        if token spelled incorrectly then
          suggestions = spellcorrect(token)
          if suggestions is not empty then
            push tokenize(first(suggestions)) into list
          end if
        end if
      end if
    end if
  end if
end while
```

A similar procedure is repeated to retrieve SentiWordNet scores. The lookup in pre-trained GloVe is substituted with a lookup in SentiWordNet.

3.1 Results

Fourfold cross-validation is used to measure the accuracy. To reduce the effects of random initialization on accuracy, each algorithm is run twice, and the average accuracy is reported in Table 2. The sentences were classified into two classes.

3.2 Discussion

The best traditional classifier for this dataset is the ridge regression classifier, giving an accuracy of **80.7%**. Using one-hot vectors matched that accuracy at 80.6%, showing that LSTM managed to find patterns in the very sparse sequence of vectors. Using LSTM with SentiWordNet scores achieved 69.4% accuracy, better than pure chance of 50%, showing that SentiWordNet scores do contain information. Using LSTM with GloVe vectors resulted in better accuracy, at 85.3%, than the best traditional classifier. Comparing with one-hot vectors, we conclude that the semantic information that GloVe vectors contain helped classification. The addition of SentiWordNet scores further improved accuracy to **85.8%**. We can thus imply that semantic and sentiment information improved our classifier.

4 Notes dataset

Four classes of 20 notes each are used in the experiment, totally 80 notes. The 20 suicide notes and 20 hoax notes are obtained from Cincinnati Hospital Medical Centre. The

Table 2 Classification accuracy on UCI sentiment labelled sentences

Classifier	Accuracy/%
Naíve Bayes Multinomial	77.5
Naíve Bayes Bernoulli	78.2
Linear SVC with L1-based feature selection	79.5
Linear SVC with L1 penalty	79.8
Linear SVC with L2 penalty	80.2
K-nearest neighbour	72.4
Rocchio	75.9
Ridge regression	**80.7**
Passive aggressive	75.7
Perceptron (Linear SGD without penalty)	74.6
Linear SGD with L1 penalty	78.9
Linear SGD with L2 penalty	79.4
Linear SGD with elastic net penalty	79.3
LSTM using one-hot vectors	80.6
LSTM using SentiWordNet scores	69.4
LSTM using GloVe vectors	85.3
LSTM using GloVe vectors⌒SentiWordNet scores	**85.8**

hospital has de-identified the notes to protect the identity of the patients and the deceased before allowing us access to them. We consider suicide notes to be written by people who died in their suicide attempt, and hoax notes to be written by people who did not. The 20 positive and neutral notes are chosen based on rating by 12 undergraduates in a pre-study. This dataset is pre-processed to correct spelling, punctuation and spacing inconsistencies.

In addition to the notes, we collected valence–arousal (VA) rating for each note. Twelve healthy participants are tasked to rate a subset of 20 notes each on the valence and arousal scale. The notes are arranged in random order. Each scale ranges from –5 to 5 to simplify data input, yielding an approximation of the Circumplex model. The selection of the subset of notes is such that each note would have 3 ratings.

4.1 Parameters

Documents were tokenized into words using the `PTBTokenizer`. To obtain document vectors, the tokens were stemmed using Porter stemmer, and then, the tf·idf statistic was computed. To obtain one-hot vectors, the tokens were similarly stemmed. The parameters used for the classifiers were

K-nearest neighbour:	10 nearest neighbour
Passive aggressive:	trained for 50 iterations
SGD:	trained for 50 iterations
LSTM:	trained for 28 iterations

4.2 Results

Given the small dataset, accuracy fluctuated a lot with k-fold validation. Thus, we used leave-one-out cross-validation to measure the accuracy of the classifiers. To reduce the effects of random initialization, each algorithm is run 3 times and the average accuracy is reported in Tables 3 and 4. The notes were classified into four classes.

4.3 Discussion

The best performing traditional classifier on our dataset is the Naíve Bayes Bernoulli classifier, with accuracy of

Table 3 Classification accuracy on notes dataset

Classifier	Accuracy/%
Naíve Bayes Multinomial	56.2
Naíve Bayes Bernoulli	**71.3**
Linear SVC with L1-based feature selection	58.8
Linear SVC with L1 penalty	67.5
Linear SVC with L2 penalty	66.2
K-nearest neighbour	43.8
Rocchio	70.0
Ridge regression	62.1
Passive aggressive	59.6
Perceptron (Linear SGD without penalty)	58.8
Linear SGD with L1 penalty	59.2
Linear SGD with L2 penalty	64.2
Linear SGD with elastic net penalty	65.0
LSTM using one-hot vectors	55.8
LSTM using SentiWordNet scores	38.3
LSTM using GloVe vectors	*70.4*
LSTM using GloVe vectors and VA	**73.8**
LSTM using GloVe vectors⌒SentiWordNet scores	*72.1*
LSTM using GloVe vectors⌒SentiWordNet scores and VA	*72.1*

Table 4 Prepending/appending VA to sequence data

Classifier	Accuracy/%
LSTM with GloVe vectors, then VA	67.5
LSTM with GloVe vectors⌒SentiWordNet scores, then VA	68.8
LSTM with VA, then GloVe vectors	74.2
LSTM with VA, then GloVe vectors⌒SentiWordNet scores	75.0

Table 5 Comparison with recent methods

Classifier	Accuracy (UCI)/%	Accuracy (Notes)/%
BiGRU [17]	71.1	68.3
Big data-based deep learning [16]	68.2	66.0
Deep classifier [9]	71.3	54.5

71.3%. Using LSTM with GloVe vectors only performed almost as good at *70.4%*. Comparing LSTM using one-hot vectors and LSTM using GloVe vectors, we can see that the pre-trained GloVe vectors brought in additional semantic relationship information that helped in classification, bringing accuracy 14.6% higher. Most recent deep learning-based methods using only word vector achieve 71.3% on UCI and 68.3% on notes dataset. The results are presented in Table 5.

In LSTM using SentiWordNet scores, it can be seen that emotional scores alone, without any word vectors, does carry enough information to classify with 38.3% accuracy, better than pure chance of 25%. The addition of SentiWordNet scores to LSTM using GloVe vectors increases accuracy to *72.1%*. Further addition of valence–arousal ratings did not improve accuracy, showing that both SentiWordNet and valence–arousal provided similar information. Interestingly, using just valence–arousal ratings with GloVe vectors yields a better result of **73.8%**. A simpler network might have enabled the neural network to learn to leverage the information from the ratings better. Presenting the valence–arousal ratings first and appending SentiWordNet scores both increased accuracy with the formal being more effective. The accuracy of 74.2% and 72.1% both outperformed the best traditional classifier.

However, presenting the valence–arousal ratings last reduced accuracy instead (reduced *70.4%* to 67.5%; reduced *72.1%* to 68.8%). Looking at value produced by the loss function as the training progresses, we observed that the loss decreases slower and hits a plateau at a higher loss than presenting the valence–arousal first. We suspect it is due to have the 300-dimension GloVe vectors being 3 time steps further away from the output, and the LSTM had to assign several nodes to the role of shutting of the input gates of other nodes to preserve the values in the memory cells. This left less nodes for the role of learning the pattern between the input and the output.

Presenting the valence–arousal ratings first and appending SentiWordNet scores at the same time achieved the best accuracy, yielding **75%**. Intuition would be that our documents are emotionally charged, and thus, attaching emotional scores would bring documents from different classes further from each other, making classification easier.

5 Conclusion

This research aims to improve classification beyond what classic text classification algorithms offer. It achieved the goal through breaking the tradition of treating document as a vector. Instead, each document was represented as a sequence of vectors. The main novelty of or work is to use word-based and sentiment-based encoding for deep learning-based text classification. Doing so preserved the position of words in the document, while giving the flexibility of incorporating semantic information from GloVe vectors, valence–arousal ratings and sentiment information from SentiWordNet. Solving the problem might involve using non-homogeneous nodes in a layer, using skip-layers, or changing the LSTM memory block. We would also like to investigate what other kinds of data can be incorporated into our model.

The main drawback of the proposed system is the sparse representation of the words. Sometimes, this comes with information loss and performs poorly. Our method can be improved by utilizing more powerful sentiment vectorization method and use of advance classifiers.

Funding Open access funding provided by UiT The Arctic University of Norway (incl University Hospital of North Norway).

Declarations

Conflict of interest Authors declare no conflict of interest.

References

1. Kowsari K, Meimandi KJ, Heidarysafa M, Mendu S, Barnes L, Brown D (2019) Text classification algorithms: a survey. Information 10(4):150
2. Mendez JR, Cotos-Yanez TR, Ruano-Ordas D (2019) A new semantic-based feature selection method for spam filtering. Appl Soft Comput 76:89–104
3. Jauhiainen T, Lui M, Zampieri M, Baldwin T, Lindén K (2019) Automatic language identification in texts: a survey. J Artif Intell Res 65:675–782
4. Chen F, Huang Y (2019) Knowledge-enhanced neural networks for sentiment analysis of chinese reviews. Neurocomputing 368:51–58
5. Shuo X (2018) Bayesian naïve bayes classifiers to text classification. J Inf Sci 44(1):48–59
6. Al-Khurayji R, Sameh A (2017) An effective arabic text classification approach based on kernel naive bayes classifier. Int J Artif Intell Appl 8(6)
7. Hadi W, Al-Radaideh Qasem A, Alhawari S (2018) Integrating associative rule-based classification with naïve bayes for text classification. Appl Soft Comput 69:344–356
8. Eminagaoglu M (2020) A new similarity measure for vector space models in text classification and information retrieval. J Inf Sci, page 0165551520968055
9. Mohammad AH, Alwada'n T, Al-Momani O (2016) Arabic text categorization using support vector machine, naïve bayes and neural network. GSTF Jal Comput (JoC), 5(1): 108
10. Prasetijo Agung B, Isnanto RR, Eridani D, Soetrisno Yosua AA, Arfan M, Sofwan A (2017) Hoax detection system on indonesian news sites based on text classification using svm and sgd. In: 2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pages 45–49. IEEE
11. BaygIn M (2018) Classification of text documents based on naive bayes using n-gram features. In: 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), pages 1–5. Ieee
12. Elghazel H, Aussem A, Gharroudi O, Saadaoui W (2016) Ensemble multi-label text categorization based on rotation forest and latent semantic indexing. Expert Syst Appl 57:1–11
13. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NAE, Arshad H (2018) State-of-the-art in artificial neural network applications: A survey. Heliyon 4(11):e00938
14. Abiodun OI, Jantan A, Omolara AE, Dada KV, Umar AM, Linus OU, Arshad H, Kazaure AA, Gana U, Kiru MU (2019) Comprehensive review of artificial neural network applications to pattern recognition. IEEE Access 7:158820–158846
15. Jiang D, He J (2021) Text semantic classification of long discourses based on neural networks with improved focal loss. Comput Intell Neurosci 2021
16. Chatterjee A, Gupta U, Chinnakotla MK, Srikanth R, Galley M, Agrawal P (2019) Understanding emotions in text using deep learning and big data. Comput Hum Behav 93:309–317
17. Xinxin L, Zhang H (2021) Sentiment analysis method of network text based on improved at-bigru model. Scientif Program 2021
18. Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2016) Lstm: a search space odyssey. IEEE Trans Neural Netw Learn Syst 28(10):2222–2232
19. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. Proc Empiric Methods Nat Language Process EMNLP 2014 12:1532–1543
20. Baccianella S, Esuli A, Sebastiani F (2010) Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In: Lrec, volume 10, 2200–2204
21. Apache Software Foundation. Class standardanalyzer. 2015(4 Dec), 10 Jun 2015 2015. URL https://lucene.apache.org/core/5_2_1/analyzers-common/org/apache/lucene/analysis/standard/StandardAnalyzer.html
22. The Stanford Natural Language Processing Group. Stanford tokenizer. 2015(7 Nov), 5 Sep 2015 2015. URL http://nlp.stanford.edu/software/tokenizer.shtml
23. Porter MF (1980) An algorithm for suffix stripping. Program 14(3):130–137
24. The Stanford Natural Language Processing Group. morphy - discussion of wordnet's morphological processing. 2015(18 Nov), 28 Oct 2010 2010. URL http://wordnet.princeton.edu/wordnet/man/morphy.7WN.html
25. Shah K, Patel H, Sanghvi D, Shah M (2020) A comparative analysis of logistic regression, random forest and knn models for the text classification. Augment Human Res 5(1):1–16
26. Venkatachalam K, Balakrishnan S, Prabha R, Premnath SP (2018) Effective feature set selection and centroid classifier algorithm for web services discovery. Int J Pure Appl Math 119(12):1157–1172
27. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780 (**ISSN 0899-7667**)
28. Greff K, Srivastava RK, Koutník J, Steunebrink Bas R, Schmidhuber J (2015) Lstm: A search space odyssey. *arXiv preprint*arXiv:1503.04069
29. Zaremba W, Sutskever I, Vinyals O (2014) Recurrent neural network regularization. *arXiv preprint*arXiv:1409.2329
30. Dieleman S, Schlüter J, Raffel C, Olson E, Sønderby SK, Nouri D, Maturana D, Thoma M, Battenberg E, Kelly J, De Fauw J, Heilman M, diogo149, McFee Brian, Weideman Hendrik, takacsg84, peterderivaz, Jon, instagibbs, Dr. Kashif R, CongLiu, Britefury, Degrave J (2015) Lasagne: First release. https://doi.org/10.5281/zenodo.27878
31. Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, Turian J, Warde-Farley D, Bengio Y. Theano: A cpu and gpu math compiler in python. In: Proc. 9th Python in Science Conf, pages 1–7
32. Pennington J, Socher R, Manning Christopher D (2014) Glove: Global vectors for word representation. 2015(8 Oct). URL http://nlp.stanford.edu/projects/glove/
33. Princeton University. About wordnet. 2010. URL http://wordnet.princeton.edu
34. Baccianella S, Esuli A, Sebastiani F. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In: LREC, volume 10, pages 2200–2204
35. Posner J, Russell JA, Peterson BS (2005) The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. Dev Psychopathol 17(03):715–734 (**ISSN 1469-2198**)
36. Kotzias D, Denil M, De Freitas N, Smyth P. From group to individual labels using deep features. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 597–606. ACM. ISBN 1450336647