Journal of Ambient Intelligence and Humanized Computing (2021) 12:10267–10287
https://doi.org/10.1007/s12652-020-02800-7

ORIGINAL RESEARCH

Bidirectional transfer learning model for sentiment analysis of natural language

Shivani Malhotra[1] · **Vinay Kumar**[1] · **Alpana Agarwal**[1]

Received: 29 April 2020 / Accepted: 5 December 2020 / Published online: 2 January 2021

Abstract
The contemporary unsupervised word representation methods have been successful in capturing semantic statistics on various Natural Language Processing tasks. However, these methods proved to be futile in addressing tasks like polysemy or homonymy, which prevail in such tasks. There has been a rise in the number of state-of-the-art transfer learning techniques bringing into play the language models pre-trained on large inclusive corpus. Motivated by these techniques, the present paper proposes an efficacious transfer learning based ensemble model. This model is inspired by ULMFit and presents results on challenging sentiment analysis tasks such as contextualization and regularization. We have empirically validated the efficiency of our proposed model by applying it to three conventional datasets for sentiment classification task. Our model accomplished the state-of-the-art outcomes remarkably when compared to acknowledged baselines in terms of classification accuracy.

Keywords Universal language model fine-tuning (ULMFit) · Bidirectional encoder representations from transformers (BERT) · Average stochastic gradient weight-dropped LSTM (AWD-LSTM) · Transfer learning · Sentiment classification

1 Introduction

Natural Language Processing (NLP) is an area of artificial intelligence which enables computers to read, understand and process human language. It is a discipline specifically focused on building a relationship between natural language data and data science. The natural language data generated from conversations, videos, speeches, image captions, etc is unstructured in nature as it is not in the form to be put into conventional row and column arrangement of a database. Natural Language Processing is a set of algorithms and techniques for extracting meaningful information from data by adding necessary structure to it (Collobert et al. 2011). It can be divided into following three different sets of approaches:

1. *Rule Based* Bird et al. (2009) stated that rule-based systems perform sentiment analysis using hand-crafted set of rules. As these systems are based on linguistic structures resembling human way of building grammar, they tend to focus on decoding the linguistic relationships between words to interpret the polarity of piece of text. The rule based system approach comprises of a definite set of rules employing traditional NLP approaches such as part-of-speech tagging, stemming, tokenization, lexicons, etc.
2. *Statistics Based* Manning et al. (1999) explained that statistical approach does not rely on the hand-crafted rules but traditional machine learning perspective including probabilistic modeling, likelihood maximization, linear classifiers etc. This approach is based on statistical models like Hidden Markov Models, Perceptrons, Logistic Regression, etc. It takes the sentiment task as a classification problem and feeds it to a classifier. Statistical hypothesis in general is based on the data generated in accordance with some unknown probability distribution and makes inference out of it. The classifier predicts the most probable outcome from the distribution as the corresponding label.
3. *Neural Networks Based* Neural Networks are designed to find generalized predictive patterns as they are not based on hypothesis about the correlation among the variables. According to Young et al. (2018), this is the biggest advantage of neural networks over the traditional NLP techniques. Feature engineering is skipped in these net-

✉ Shivani Malhotra
smalhotra_phd18@thapar.edu

[1] Department of Electronics and Communication Engineering, Thapar Institute of Engineering and Technology, Patiala, India

works as they automatically learn to capture the important features of data (Jean-François 2017). Some specific neural networks employed in NLP comprise of Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Long-Short Term Memory (LSTM), etc. These state-of-art networks (Bouazizi and Ohtsuki 2017; Chen et al. 2017) find applications in many areas such as text analytics, sentiment analysis, speech recognition, financial trading, etc.

For computers to precisely comprehend the unstructured data, it is necessary for them to understand the sentiments or contextual meanings hidden in the natural language (Collobert et al. 2011). Present paper focuses on this abstract idea known as sentiment analysis.

Sentiment analysis (Pathak et al. 2020) refers to automatically analyzing the sentiments and categorizing them into certain classes. This concept can be categorized into three major levels: aspect level, sentence level and document level.

- Aspect level sentiment analysis is a text mining technique that divides the text into different attributes and allots each attribute a sentiment class (Gupta et al. 2019).
- Sentence level sentiment analysis involves dividing the sentence into two classes: subjective and objective (Gupta et al. 2019). The subjective part comprises of emotions, beliefs, views, etc; whereas the objective part consists of the fact-based data.
- Document level sentiment analysis (Wu et al. 2020) considers the whole document as a fundamental data block. It focuses on extracting sentiment(s) from the document to determine the overall opinion of document or any of its particular entity (Gupta et al. 2019).

With the proliferation of social media, multimodal sentiment analysis is set to bring new opportunities with the arrival of complementary data streams for improving and going beyond text-based sentiment analysis.

Soleymani et al. (2017) illustrated that sentiment analysis techniques can be applied to any mode of data such as audio, video or text. The proposed work utilizes automated analysis techniques for text data at sentence level to detect the polarity of labelled text solely based on its content. The experimental results are obtained from three datasets which help in testing the overall effectiveness of our model.

The comprehensive contributions of this paper are as follows:

1. We propose a bidirectional model inspired from ULMFit which is based on transfer learning.
2. We have demonstrated the experimental results of two important aggregation architectures used for extracting features from embedding sequences namely Concat Pool and Attention, with and without zeta. When zeta is incorporated into the model, it reinforces the performance of both aggregation architectures relative to the case when not utilized. It has also been observed that irrespective of the scaling parameter, Concat pool has performed efficiently relative to Attention mechanism.
3. We have reckoned that our proposed solution is an excellent and novel substitute of Attention mechanism. Also, we have not confined the comparison to this mechanism only, we have compared our proposed model with other state-of-the-art deep and non-deep learning models also.

The organization of the present paper is as follows: Sect. 2 provides an insight of the previous work of some researchers related to the paper; Sect. 3 lists out the datasets used in the research work along with their detailed study; Sect. 4 looks into some of the index terms used in the research work to get better understanding; Sect. 5 introduces the proposed model and provides the experimental results; Sect. 6 shows the experimental results of our proposed model and its comparison with other state-of-the-art models; Sect. 7 abridges the paper and provides suggestions for future work.

2 Literature survey

There has been a lot of research in the field of sentiment classification, making it one of the most popular tasks in NLP (Cambria 2016). The advancements include traditional methods, deep learning methods, transfer learning methods, etc based on different sentiment classification algorithms. As stated by Wang et al. (2020) and Mirończuk and Protasiewicz (2018), these methods have evolved over the time to classify sentiments accurately in accordance with their contextual meanings. Present section discusses some of the significant NLP approaches used for sentiment classification.

2.1 Traditional methods for sentiment analysis

In recent years, several machine learning techniques based on shallow models such as Support Vector Machine, Logistic regression, etc (Jiang et al. 2018) are employed for sentiment analysis. These models are trained on a limited number of hand crafted features. Also, the features have to be identified by a domain expert in order to reduce the data complexity and provide an ease for these traditional techniques to work. Collobert et al. (2011) explained that most state-of-art techniques in various NLP tasks, such as Named-Entity Recognition (NER), Part-of-Speech (POS) tagging, Semantic role labelling are outperformed by a simple deep learning model. For modeling tedious NLP tasks, statistical NLP turned out to be one of the primary options (Haddoud et al. 2016; de Araujo et al. 2020). However, it often used to

suffer from curse of dimensionality in its initial phases while learning joint probability functions of language models, as stated by Bengio et al. (2003). This concept gave motivation for learning distributed representations of words present in low-dimensional space. Mikolov et al. (2013b) proposed novel architectures for producing high quality word representations, thereby producing more accurate results compared to the ones obtained from the traditional techniques.

2.2 Deep learning methods for sentiment analysis

Distributional vectors follow distributional hypothesis, which state that words with similar semantics are likely to occur in similar context. The semantic similarity is directly proportional to the distance between vectors (Mikolov et al. 2013b). The distance between these vectors is measured using cosine similarity.

Word embeddings can be pretrained in such a way that they capture the semantic information and give predictions based on the contextual meanings of words. These embeddings, due to their low dimensionality, have proved to be fast and efficient in capturing the conceptual semantics (Cambria et al. 2017; Turney and Pantel 2010). In the recent years, models creating these word embeddings have been facile. Thus, there was no need for deep learning models (Rane and Kumar 2018; Saif et al. 2016) to create quality embeddings. However, it is known that word embeddings are responsible for state-of-art results in NLP tasks in which words, phrases, etc are represented using these embeddings (Jianqiang and Xiaolin 2017). For instance, Glorot et al. (2011) employed embeddings along with stacked autoencoders for domain adaptation in sentiment analysis whereas Hermann and Blunsom (2013) learnt the composition of sentence by introducing CCAE (Combinatory Categorial Autoencoders). Developments in NLP over these years marked the foundation of research in distributed representations.

Mikolov et al. (2013a) developed a method to create word embeddings known as word2vec. It includes two models namely Common bag of words (CBOW) and Skip Gram which are used to obtain word embeddings. In CBOW architecture, distributed representations of words surrounding the input word are combined to predict the current target word. On the other hand, skip gram architecture usually attains the opposite of what CBOW does. It predicts the surrounding words based on the given target word. Pennington et al. (2014) showed the above mentioned word2vec methods do not utilize the corpus statistics efficiently. It happens because they train on different restricted context windows and not global co-occurrence information. They proposed a new model called GloVe, which directly captures the global representation of data. GloVe focuses on the global word-to-word co-occurrence matrix, which relates word embeddings to the co-occurrences of words over the whole corpus.

In 2016, Facebook developed fastText; an extension of vanilla word2vec model. It takes word parts into account and learns representations for sub-words, also known as character n-grams. Bojanowski et al. (2017) explained how fastText works by taking morphology into consideration; this technique strengthens learning in extremely inflected languages. The static embeddings came up with some limitations as discussed in Sect. 4.1. As a result, they fail to capture higher-level information.

2.3 Transfer learning methods for sentiment analysis

For instance, McCann et al. (2017) utilized an encoder of a supervised neural machine conversion to bring context out of the word embeddings. Eventually, the context based word vectors are concatenated with the pretrained word embeddings. On the other hand, Neelakantan et al. (2015) conducted research by training each vector for individual word senses, which means multiple embeddings per word were learned. With these techniques, the issue of missing context was eliminated. These techniques, however, could not eradicate the need of training the factual task model from scratch.

Many machine learning algorithms have an assumption that the training and future data must have same dimensions and be in same vector space (Pan and Yang 2009). However, this does not hold in real life applications. This aforesaid assumption was solved by a concept in which knowledge gained in one domain of interest is transferred to another. Therefore transfer learning emerged as a new learning strategy for developing deep learning models (Liu et al. 2019).

Peters et al. (2018) introduced deep contextualised word representations known as ELMo (Embeddings from Language models), capable of modeling complex characterstics of semantics. They proved that ELMo bi-LSTM can be trained on a large dataset and then used as an essential part in other NLP models. The word vectors allocated to a token or word are a learned function of inner states of language model. In other words, it is context dependent and same word can have different vectors under different contexts. This method trains only the main task model from scratch and considers pre-trained embeddings as a fixed parameter, thereby restricting its success.

Howard and Ruder (2018) proposed ULMFit (Universal Language Model Fine-tuning) model. It recommends language modeling on a massive corpus and uses this language model as a backbone for fine-tuning the classifier. The architecture used by ULMFit for the language modeling is AWD-LSTM which stands for Average Stochastic Gradient Descent Weight Dropped LSTM. To apply this concept to other datasets, the parameters of language model have to be fine-tuned and a classifier layer has to be appended and trained. To delve into contextualization, an intuitive way to

find the alignment between the words was introduced by Bahdanau et al. (2015). This is performed by visualizing the weights analogous to annotations.

Peters et al. (2017) extracted contextual features from right-to-left and left-to-right from a deep learning model based on LSTM. But Devlin et al. (2019) developed an attention based model called BERT (Bidirectional Encoder Representation from Transformers) for training deep bidirectional representations. This architecture can extract contextual features from right-to-left, left-to-right and combined left-to-right, thereby allowing high level of parallelism.

3 Datasets

Sentiment detection reckons on text data. For experiment, we have chosen three databases commonly used in sentiment analysis nowadays: IMDb (Maas et al. 2011), US Airline Twitter (Crowdflower 2016), Real Life Deception Detection (Pérez-Rosas et al. 2015). A brief description of datasets is presented below.

1. *IMDb* It is a movie review dataset containing 50,000 movie reviews for binary sentiment classification. Each sample in this dataset is a text document. They are combined to form training and test files. It comprises of binary reviews namely positive and negative as shown in the Table 1 below.
2. *US Airline Twitter* This dataset was first released by Crowdflower in 2015 comprising of tweets on major US Airlines such as United, US Airways, Southwest, Delta and Virgin America. The tweets have been classified into 3 categories: Positive, Negative and Neutral. The present paper focuses on tweets and their labels for sentiment classification as shown in the Table 1 below.
3. *Real Life Deception Detection (RLDD)* It is multimodal dataset comprising of real life videos of courtroom trials. It is an aggregation of both text and visual data bisected into: Deceptive and Truthful. The videos have been sourced from various Youtube channels. We have used the transcripts of the courtroom trials to build the required dataset. A text document is created using both truthful and deceptive cases, refer Table 1 for the details.

4 Index terms

4.1 Word embeddings

According to Le and Mikolov (2014), the concept of word embeddings has been one of the outstanding developments in the field of Natural Language Processing in the last few years. These are also known as distributed representations of text in n-dimensional space. Word embeddings primarily bridge the human understanding of language to that of a machine. These are a set of strategies in which tokens are mapped to real valued vectors in a meaningful vector space. The distance between vectors is directly proportional to semantic similarity between the tokens. These vectors values are learned in such a way that they resemble a neural network.

There are many word embeddings used by researchers nowadays, but the present paper focuses on two kinds of pre-trained embeddings namely GloVe (Pennington et al. 2014) and fastText (Joulin et al. 2017). We selected these embeddings as they are pre-trained on a large corpus and can be employed in diverse downstream tasks such as named entity recognition, part-of-speech tagging, language modeling, etc.

GloVe

To the full extent, GloVe stands for Global Vectors for word representation. It is an unsupervised learning algorithm developed by Pennington et al. (2014). GloVe finds its prime application in generating word embeddings based on the co-occurence information of words present in the corpus. The process of building the word embeddings in GloVe

Table 1 A brief outline of statistics of experimental datasets

Dataset	Types of samples		Total number of samples	Training examples	Test examples
IMDb Movie Review	Positive/ **Negative**	25,000/ **25,000**	50,000	40,000	10,000
Twitter	Positive/ **Negative**/ *Neutral*	2363/ **9178**/ *3099*	14,640	11,712	2928
Real Life Deception Detection	Truth/ **Deceptive**	60/ **61**	121	97	24

is carried out by training the collection of global word to word co-occurrence statistics from the corpus. The resulting embeddings exhibit intriguing one dimensional substructure of vector space. The present paper uses GloVe embeddings of 300 dimensions containing 840 billion tokens for the sentiment classification purpose.

fastText

GloVe embeddings made remarkable developments in the field of NLP, but they were not able to generalize for the words not present in the GloVe embeddings. In 2016, Facebook's AI research team released a library for learning word embeddings and performing sentiment classification known as fastText. According to Joulin et al. (2017), this library helps to build an unsupervised or supervised learning algorithm to obtain word vectors using a deep learning model. Their results presented major improvement of fastText when compared to other embeddings such as word2vec, GloVe, etc. This improvement refers to dividing the word into sub-words known as n-grams (where n is total number of sub-words). The building blocks for sentiment classification in this case are sub-words or n-grams and not words or sentences. This makes way for generalization as long as characters of words present in GloVe embeddings are present in our database too.

Although the pre-trained embeddings immensely influence research in NLP, they still show following major limitations:

- Static embeddings presume that a word has same interpretation across the corpus or they do not support polysemy. Due to the limitation, contextual meaning of the word is lost (Wang et al. 2020).
- When the embeddings are loaded into the model, only embedding layer is trained and not the hidden layers (Yosinski et al. 2014). Changes made in the embeddings do not reflect in hidden and output layers of the model. As a result, these layers have to be trained from scratch. Thus, pre-trained embeddings fail to capture higher-level information.

As a result of above mentioned limitations, researchers felt the need of a trained model capable of capturing contextual meanings of words present in the corpus. They introduced a technique known as Transfer Learning, wherein models are pre-trained on large corpus and fine-tuned for specific NLP tasks. The knowledge gained in pre-training step is utilized for fine-tuning on the target task. The strength of this technique includes accelerating the training time of a model by reusing the modules of hitherto developed models. These trained models are open-sourced and prove to be successful in various NLP classification challenges (Liu et al. 2019; Zheng et al. 2020).

4.2 Pre-trained models

ULMFit

Howard and Ruder (2018) proposed that transfer learning is the backbone of a pre-trained model, ULMFit, which stands for Universal Language Model Fine Tuning. ULMFit has been completely executed in *fastai* library, which simplifies training of fast and accurate neural nets using modern best practices. A brief outline of ULMFit is presented below.

1. *Language pre-training on general domain corpus* Text data used in the model has to be tokenized and encoded. These unique tokens are then prioritized based on the usage in an order from most often to least often used. The text data is fed to embedding layer of the model. The embedding matrix in the model has vectors for each token in the general domain corpus named WikiText-103. The encoded tokens present in the corpus are matched to their corresponding vectors by using one-hot encoding. After the embedding layer, text data enters three stacked LSTM layers known as Average Stochastic Gradient Weight-Dropped LSTM (AWD-LSTM), where the model training takes place. The output of embedding layer is in form of tensors with word embeddings. These tensors are fed as inputs to first LSTM layer. They are subsequently passed to the second and third LSTM layers. The hidden state of last LSTM layer has same shape as of embedded input. Softmax function is applied to the output of last LSTM layer to get corresponding probabilities. The first step is computationally expensive; which is why *fastai* has made this model publicly available (Howard and Ruder 2018).
2. *Language Model fine-tuning on target task* The pre-trained model is fine tuned on the target task at this step. To achieve this, the target dataset has to be pre-processed and AWD-LSTM rebuild in order to load the weights of the language model. Also, the embeddings have to be adjusted according to the target dataset before recalliberating AWD-LSTM and loading the weights. However, the recurrent connections of an LSTM are usually susceptible to overfitting. If dropouts similar to LSTM's hidden state are applied, its capability to hold on to long-term dependencies is challenged.

   To resolve this problem, an alternative to dropout was derived by researchers known as DropConnect (Merity et al. 2018). Unlike the classical concept of dropout, where random subset of activations are set to zero, DropConnect selects random subset of weights and sets them

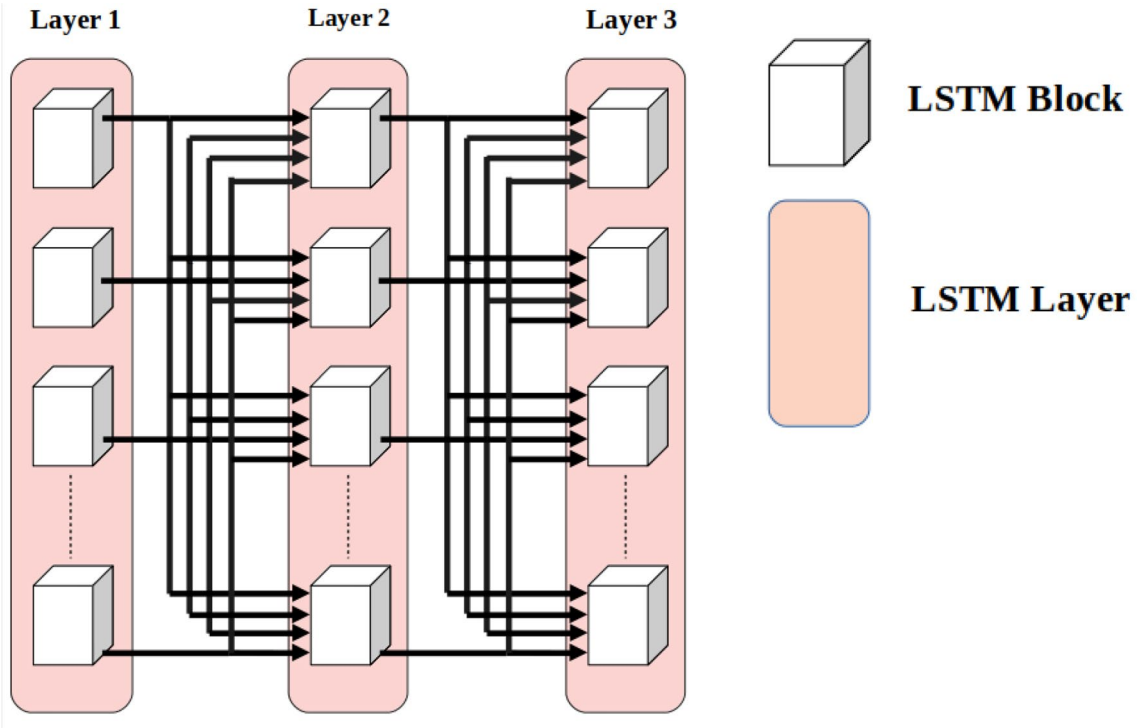

Fig. 1 An arrangement of three stacked interconnected LSTM layers containing multiple LSTM blocks constitute a neural network. Nodes of this neural network are shown by LSTM blocks and the weights by their interconnections

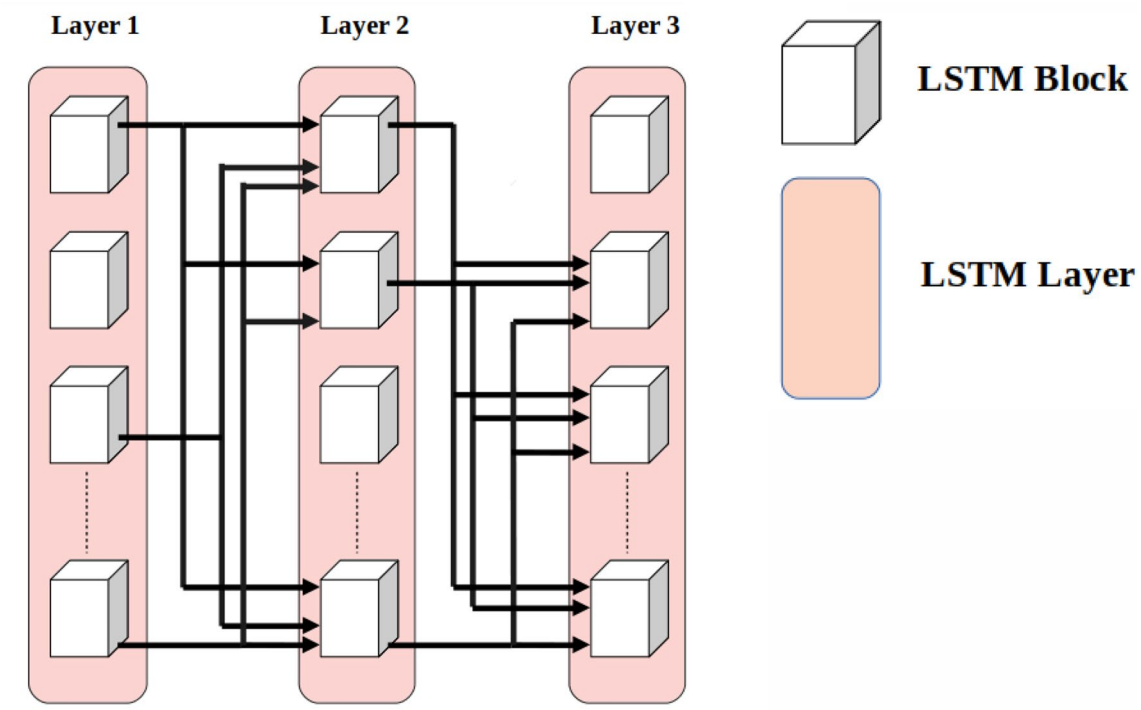

Fig. 2 Dropout method applied to a neural network where random subset of activations are set to zero

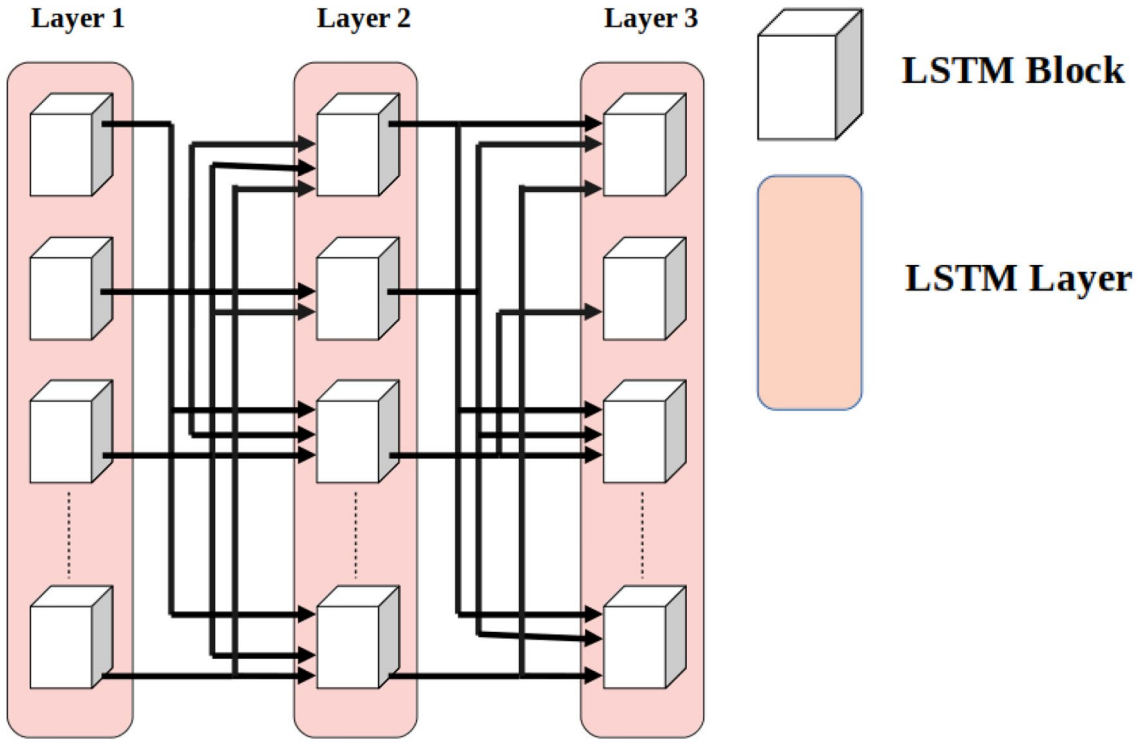

Fig. 3 DropConnect method applied to a neural network where random subset of weights are deactivated

to zero. With this, input coming from random subset of units in the previous layer is received by each unit. Figures 1, 2 and 3 explain the difference between dropout and DropConnect when compared to a standard neural network using three LSTM layers. The LSTM blocks inside these layers are activations for the neural network.

The mathematical formulation of an LSTM can be expressed as set of subequations (a)–(f) of Eq. (1) as:

$$\begin{aligned} i_t &= \sigma(W^i x_t + U^i h_{t-1}) && \text{(a)} \\ f_t &= \sigma(W^f x_t + U^f h_{t-1}) && \text{(b)} \\ o_t &= \sigma(W^o x_t + U^o h_{t-1}) && \text{(c)} \\ \tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1}) && \text{(d)} \\ c_t &= i_t \odot \tilde{c}_t + f_t \odot + \tilde{c}_{t-1}) && \text{(e)} \\ h_t &= o_t \odot \tanh(c_t) && \text{(f)} \end{aligned} \tag{1}$$

where $[W^i, W^f, W^o, U^i, U^f, U^o]$ are weight matrices, x_t is the vector input to the timestep t, h_t is the current exposed hidden state, c_t is the memory cell state, and $\odot$ is the element-wise multiplication. Merity et al. (2018) proved that DropConnect is applied to hidden-to-hidden weight matrices $[U^i, U^f, U^o]$ instead of hidden or memory states, thereby preventing overfitting on the recurrent connections of the LSTM. Howard and Ruder (2018) explained that training the entire model at once leads to catastrophic forgetting in the three LSTMs as they still comprise of old weights from the pre-trained language model. In this situation, it is necessary to 'freeze' the weights of stacked LSTM layers while the embedding and output layers are trained so that they can be adjusted to the LSTM weights. After this, all the AWD-LSTM layers are unfreezed for fine-tuning. This is how the language model learns task specific features of the language.

3. *Target Task Classification* Sentiment classification is performed by adjusting the language model architecture. To achieve this, two linear functional blocks namely ReLU and softmax activations respectively are appended to the stacked LSTMs. The resulting sentiment classification of text is obtained as probabilities from the softmax layer.

Attention in ULMFit

Soft attention is convenient because it addresses the entire input state space and keeps the model fully differentiable and deterministic (Vaswani et al. 2017; Xu et al. 2015). It lets the user decide how much attention should be paid to each and every token using cosine similarity. This is performed by representing the tokens and query in the same vector space. The cosine distance is entirely differentiable with respect to

its inputs; hence the final model comes out to be differentiable. In soft attention mechanism, the gradients can be calculated instantly as compared to hard attention where they are estimated through a stochastic process. This mechanism fits well into any existing model where gradients propagate through it and remaining layers of the neural network.

Soft attention mechanism compares the source and target states to generate attention scores, denoted by α (Bahdanau et al. 2015). These scores signify how well two words are aligned with each other at a particular position t with respect to a source sequence x and target sequence y, as shown in Eq. (2). Softmax activation is used to normalize these alignment scores, which are used to generate a context vector explained in Eqs. (3)–(4); where s and h are attention mechanism parameters. Equation (5) explains how the scores are calculated from a context vector (c_t), where W_c and b_c are weight matrices and bias to be learned.

$$\alpha_{t,i} = align(y_t, x_i) \tag{2}$$

$$\alpha_{t,i} = \frac{\exp(score(s_{t-1}, h_i))}{\sum_{i'=1}^{n} \exp(score(s_{t-1}, h'_i))} \tag{3}$$

$$c_t = \sum_{i=1}^{n} (\alpha_{t,i}, h_i) \tag{4}$$

$$score(s_t, h_i) = tanh(W_c * c + b_c) \tag{5}$$

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a technique developed by Google in 2018. BERT finds its roots from pre-training contextual representations such as ELMo (Peters et al. 2018), ULMFit, Generative Pre-Training, Semi-Supervised Sequence learning, etc. Regardless of the above mentioned representations, where the text data is observed either from left to right or right-to-left or combined left-to-right training, BERT is considered to be the first deeply bidirectional language representation. Devlin et al. (2019) innovated the concept of bidirectional training of a well known attention model named Transformer, to language modelling. Transformer comprises of two distinct functional blocks: an encoder which reads the text input; and a decoder which provides a prediction for the task. Rather than processing the tokens present in a sequence one-by-one, the attention models process the tokens in relation to all other ones present in that sequence. According to them, this kind of training is achieved through a unique approach known as Masked Language Modelling through which the language models tend to acquire an insightful sense of language context and motion.

5 Proposed model

The present manuscript proposes a new model inspired from the pre-trained model ULMFit. Our model follows the ensemble strategy of Forward and Backward language models. Both of these models are two versions of the same proposed model, as shown in Fig. 4 and discussed later.

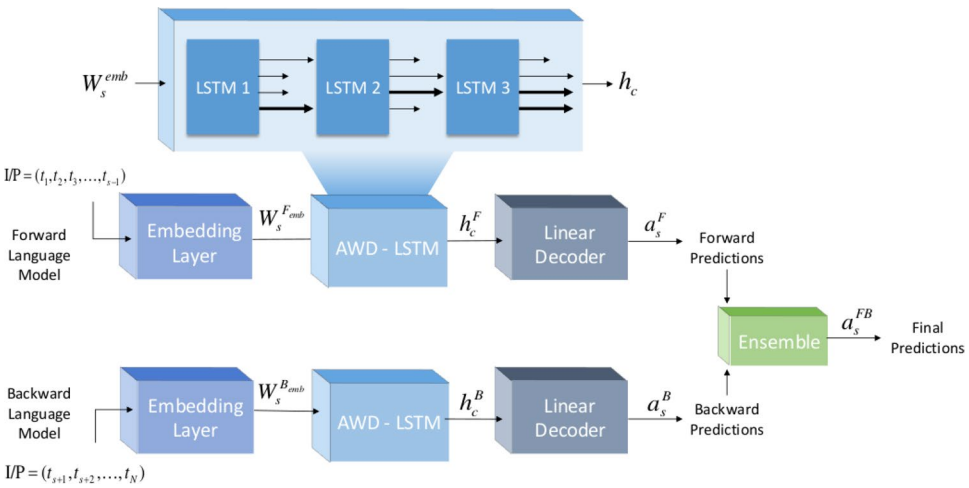

Fig. 4 Proposed model architecture: an ensemble of forward and backward language models with scaled recurrent weights of AWD-LSTM

Algorithm 1: Algorithm of our proposed model showing the necessary steps for both forward and backward language models.

Input: Token sequence: $(t_1, t_2, ..., t_N)$
Output: Probability of classifier label
1: Forward Language Model
2: Token sequence: $(t_1, t_2, ..., t_{s-1})$
3: Embedding vector: w_s^F, which is encoded by column vectors in an embedding matrix: $W_s^{F_{emb}}$
4: Hidden state as the output of the last AWD-LSTM layer is obtained as: $H_j^F = LSTM_j(H_{j-1}^F, w_s^F)$
5: Softmax function transforms all values in the decoder matrix into probabilities: $a_s^F = softmax(v^T u_s^F)$
6: Fine-tuning strategy: Discriminative fine tuning and Slanted triangular learning rates.
7: recurrent weights of AWD-LSTM scaled by a factor of zeta, denoted as 'ζ'
8: Concatenation of the last hidden state of last time step h_T with both average pooled and max pooled representations: $h_c^F = [h_T, maxpool(H_j^F), meanpool(H_j^F)]$
9: The output of the classifier is obtained as predictions in form of probabilities.
10: Backward Language Model: Repeat Steps (3) to (9) with token sequence: $(t_N, t_{N-1}, ..., t_{s+1})$
11: ensemble probabilities of both Forward and Backward Language Models: a_s^{FB}

Algorithm 1 above elaborates our proposed model in three steps. Clearly, as shown in the algorithm, we have performed both forward and backward probability modeling of a sequence of tokens. All the steps in Algorithm have been explained in Sects. 5.1 and 5.2 in detail.

5.1 Forward language model

The probability for modeling a token t_s in a token sequence at position s in the forward language model is given by Eq. (6) as:

$$p(t_1, t_2, ..., t_N) = \prod_{s=1}^{N} p(t_s | t_1, t_2, ..., t_{s-1}) \tag{6}$$

where the factor $p(* \mid *)$ is referred to as the conditional probability generated in the forward and backward token sequences.

For sentiment analysis, our language model is pre-trained on a generic massive corpus: WikiText-103 (Howard and Ruder 2018). Pre-training commences with pre-processing of data. Although there are a lot of word embeddings available for pre-processing, our language model has been trained to obtain representations for full sentences apart from the distributed representations of tokens present inside it.

For the same sequence of tokens as mentioned in Eq. (6), each word is converted into an embedding vector w_s^F, which are encoded by column vectors in an embedding matrix $W_s^{F_{emb}}$. The embedding of tth word in the vocabulary is corresponded by each column. The databunch object in *fastai* does the pre-processing in the background. The sequence of words affect the semantics of each and every word due to its dependence on foregoing words. Thus, we have employed three stacked LSTM layers (AWD-LSTM) succeeding the embedding layer, where the word embeddings are fed. The hidden state H_j^F of dimension D is obtained as the output of the last AWD-LSTM layer having same shape as the embedded input as shown in Eq. (7):

$$H_j^F = LSTM_j(H_{j-1}^F, w_s^F) \tag{7}$$

where the subscript 'j' distinguishes the three stacked LSTM layers. Finally, the hidden state is multiplied with a decoder matrix. Eqs. (8), (9) illustrate the softmax function transforming all values in the decoder matrix into probabilities.

$$u_s^F = \tanh(W_s^{F_{emb}} H_j^F + b^F) \tag{8}$$

$$a_s^F = softmax(v^T u_s^F) \tag{9}$$

where $W_s^{F_{emb}} \in R^{D*D}$, v(word level latent vector) and b ∈ D, therefore $u_s^F \in R^D$ and $u_s^F \in R$.

After the first stage of pre-training, we fine-tune the pre-trained forward language model on task specific datasets, as mentioned in Sect. 3. Two strategies are used for fine tuning the language model: Discriminative fine tuning and Slanted triangular learning rates. Discriminative fine tuning is based on the fact that different layers of a neural network should possess different values of learning rates; as these layers capture different types of information (Yosinski et al. 2014). Thus, we can tune each layer with different learning rates. The stochastic gradient descent (SGD) for the model parameter θ at time step t with a learning rate of η is given by Eq. (10):

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta) \tag{10}$$

The model parameters are split into $(\theta^1, \theta^2, ..., \theta^L)$, where L is the number of layers in the neural network and θ^l contains the parameters at lth layer. Analogous to model parameters,

we obtain $(\eta^1, \eta^2, ..., \eta^L)$ for learning rate, where η^l is the learning rate of lth layer of network. Equation (11) shows the updated SGD (SGD with discriminative fine tuning) as:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta) \tag{11}$$

The Slanted triangular learning rate is based on the fact that the model should swiftly converge to an appropriate region of parameter space when the training commences. This is performed in order to adapt the model's parameters to task specific features. An annealed learning rate can not attain this behaviour. Thus, we use Slanted triangular learning rate as shown in Eq. (12), which increases the learning rate linearly in the initial phase and then decays according to the follow-up schedule.

$$\begin{aligned} cut &= \lfloor T \cdot cut\_frac \rfloor \\ p &= \begin{cases} t/cut, & \text{if t} \leq cut \\ 1 - \frac{t-cut}{cut \cdot (\frac{1}{cut\_frac} - 1)}, & \text{otherwise} \end{cases} \\ \eta_t &= \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio} \end{aligned} \tag{12}$$

where *cut* is the iteration when we advance from initial phase towards follow-up schedule, T is the number of training iterations, η_t specifies the learning rate at an iteration t, *ratio* is comparison of lowest and highest value of learning rate, p is the fraction of number of iterations which increased or decreased, *cut_frac* represents the fraction of rising learning rate.

Now, the final task is to build a classifier using the *forward_encoder* saved in the language model fine tuning step as mentioned in Eqs. (11) and (12). The classifier is also fine tuned using the two strategies mentioned above. The weights of AWD-LSTM have been scaled by a factor of zeta, denoted as 'ζ', which lies in the range (0,1]. This scaling factor generalizes the concept of DropConnect in stacked LSTM layers.

Subequations (a)–(f) of Eq. (13) show the mathematical formulation of LSTM (as seen from Eq. (1)) after scaling as:

$$\begin{aligned} i_t &= \sigma(W^i x_t + \zeta * U^i h_{t-1}) && \text{(a)} \\ f_t &= \sigma(W^f x_t + \zeta * U^f h_{t-1}) && \text{(b)} \\ o_t &= \sigma(W^o x_t + \zeta * U^o h_{t-1}) && \text{(c)} \\ \tilde{c}_t &= \tanh(W^c x_t + \zeta * U^c h_{t-1}) && \text{(d)} \\ c_t &= i_t \odot \tilde{c}_t + f_t \odot + \tilde{c}_{t-1}) && \text{(e)} \\ h_t &= o_t \odot \tanh(c_t) && \text{(f)} \end{aligned} \tag{13}$$

When ζ is active, we do not lose any kind of information. However, when ζ is not active, traditional meaning of DropConnect is justified in our model. The scaled interconnections of AWD-LSTM layers contain information which can occur anywhere in the document. Due to this, we concatenate the last hidden state of last time step h_T from $H_j^F = (h_1, h_2, ..., h_T)$ with both average pooled and max pooled representations as illustrated in Eq. (14):

$$h_c^F = [h_T, maxpool(H_j^F), meanpool(H_j^F)] \tag{14}$$

We did not fine tune all the layers of neural network concurrently, but gradually. This is carried out by gradual unfreezing of layers. The output of the classifier is obtained as predictions in form of probabilities. These steps have been repeated for each dataset.

5.2 Backward language model

Apart from the traditional forward language models, it is necessary to consider a backward language model in order to capture the future context in embeddings (Peters et al. 2017). As discussed earlier in this section, both forward and backward language models are two versions of the same proposed architecture. Hence, a backward language model can be implemented in an analogous way to forward language model.

A backward language model predicts the preceding token if future context is present. For the backward model, Eq. (15) shows the probability for modeling the token sequence mentioned in Eq. (6):

$$p(t_1, t_2, ..., t_N) = \prod_{s=N}^{1} p(t_s | t_N, t_{N-1}, ..., t_{s+1}) \tag{15}$$

For the sequence of tokens which are run in reverse by a backward language model, each word is converted into an embedding vector w_s^B. In an analogous way to forward language model, the word embeddings are produced for the sequence $(t_N, t_{N-1}, ..., t_{s+1})$. These vectors are encoded by column vectors in an embedding matrix $W_s^{B_{emb}}$.

Subsequently, the hidden state H_j^B of dimension D is obtained as the output of the last AWD-LSTM layer having same shape as the embedded input as shown in Eq. (16):

$$H_j^B = LSTM_j(H_{j-1}^B, w^B) \tag{16}$$

where the subscript 'j' distinguishes the three stacked LSTM layers. Finally, the hidden state is multiplied with a decoder matrix. The softmax function transforms all values in the decoder matrix into probabilities as shown in Eqs. (17) and (18):

$$u_s^B = \tanh(W^{B_{emb}} H_j^B + b^B) \tag{17}$$

$$a_s^B = softmax(v^T u_s^B) \tag{18}$$

where $W_s^{B_{emb}} \in R^{D*D}$, v(word level latent vector) and b $\in$ D, therefore $u_s^B \in R^D$ and $u_s^B \in R$.

After the pre-trained language model is built, it is fine tuned for target datasets, as discussed in Sect. 3. Furthermore, the same fine tuning strategies are employed as explained in Eqs. (7) and (8). A classifier is built using the *backward_encoder* saved in the language model fine-tuning step. The classifier fine tuning is also performed using the same two strategies as in forward language model. Subsequently, the scaling factor ζ is applied to oversee the weights scaling in Back propagation through time (BPTT). The reason for introducing ζ is again to extrapolate the idea of DropConnect in AWD-LSTM. Correspondingly, we concatenate the last hidden state of last time step h_T from $H_j^B = (h_1, h_2, ..., h_T)$ with both average pooled and max pooled representations as illustrated in Equ.(19):

$$h_c^B = [h_T, maxpool(H_j^B), meanpool(H_j^B)] \tag{19}$$

Thus, after the gradual unfreezing of backward language model layers, we obtain the predictions in form of probabilities. The predictions of both forward and backward language models are ensemble to get the final predictions in form of probabilities: a_s^{FB}.

To evaluate the effectiveness of our model, we have used the same set of hyperparameters mentioned in Sect. 6 across all the datasets. We have utilized 'NVIDIA Tesla P100' GPU for faster and efficient computations.

The proposed model has been described in Fig. 4. The insight of AWD-LSTM section for both language models has been explained separately. AWD-LSTM consists of three stacked LSTM layers. The arrows represent the strength of connections or weights of the layers. The size and thickness of these arrows is directly related to weights, which in turn, relates to the features extracted in AWD-LSTM layers. For instance, a thick arrow indicates unscaled recurrent weights when ζ is unity. We have use the term unscaled because ζ being unity does not alter the recurrent weights. Thus, it is considered inactive in this case. However, when ζ becomes active, it scales the recurrent weights, as represented by plain arrows. The broken arrows also indicate scaled weights in the given range. These are the extremely small weights which have higher chances of dying out in BPTT. Thus, they are eliminated using ζ which makes the model faster.

The fine grained description of AWD-LSTM layers inside our model has been displayed in Fig. 5. As we can clearly see, this figure contains 3 stacked LSTM layers. Each unit inside a layer describes an LSTM block. These interconnections between these layers, also known as their weights, are scaled by a factor of *zeta* (ζ) as explained earlier. Unscaled recurrent weights are presented using black arrows whereas scaled weights by grey arrows of variable lengths. Figures 1, 2 and 3 can be referred to comprehend the strategy behind generalization of DropConnect method.

We tried using the uni-directional language models. However, introducing the concept of bidirectionality proved to be effective. It was observed that ensemble predictions improve the model's performance. Ensembling not only reduces the variance of our model but also results in predictions that are better than any single model. Moreover, the scaling factor eliminates extremely low weights during back propagation, thereby making the model faster.

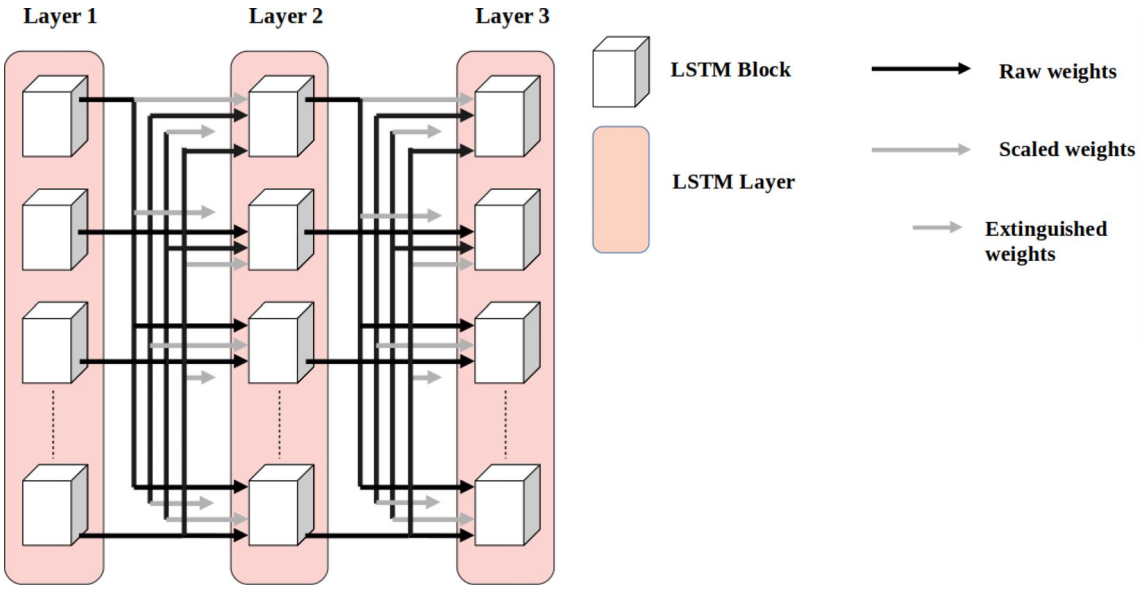

Fig. 5 Fine grained description of AWD-LSTM layers inside our proposed model showing the generalized DropConnect strategy

Table 2 Classification results of GloVe and fastText embeddings for each dataset

Dataset	Embedding	Classifier	Accuracy	Precision	Recall	F1 score
Twitter/ **RLDD**/ *IMDb*	GloVe	Logistic Regression	0.62/ **0.75**/ *0.52*	0.64/ **0.79**/ *0.54*	1.00/ **0.65**/ *0.38*	0.78/ **0.71**/ *0.45*
		Linear SVM	0.64/ **0.64**/ *0.52*	0.64/ **0.32**/ *0.50*	1.00/ **1.00**/ *1.00*	0.78/ **0.48**/ *0.67*
		Polynomial Kernel	0.64/ **0.32**/ *0.49*	0.64/ **0.32**/ *0.50*	1.00/ **1.00**/ *1.00*	0.78/ **0.48**/ *0.67*
	fastText	Hierarchical	0.67/ **0.89**/ *0.74*	0.71/ **0.89**/ *0.74*	0.71/ **0.89**/ *0.74*	0.71/ **0.89**/ *0.74*

US Airline Twitter design metrics are written as plain text, RLDD design metrics as bold values and IMDb design metrics as italic values. This formatting has been done so that parameters values can be distinguished easily

6 Experimental results

The present section compares the experimental results of proposed model with existing state-of-the-art deep and non-deep learning models. Table 2 consists of results for text classification using two word embeddings: GloVe and fastText. In case of GloVe, data was trained using biLSTM and the classifiers employed were Logistic Regression, Linear SVM and Polynomial kernel. Hierarchical classifier was employed in case of fastText embeddings.

Subsequently, we have extensively extended our research and performed experiments by taking soft attention mechanism into consideration. We tried using the consolidation of both attention layer and concat pool layer, but it performed poorly during testing than the training time. In other words we can say that it contributed to overfitting. Thus, both attention layer and concat pool layer can not be utilized together in the same model. To solve this issue, we replaced the concat pool layer with attention layer. This replacement is necessary for testing the overall performance of attention layer. We have also incorporated *drop_mult* (DM) and (ζ) in their own limits when using attention layer. The empirical results are obtained using three variations of DM and zeta which have been illustrated in Tables 3, 4, 5. Table 3 demonstrates the model's performance when ζ is not active and only weight dropout parameters are varied in their entire range. Table 4 shows results when only scaling factor ζ is active. Results when both DM and zeta are active are compiled in Table 5.

Table 3 Classification results of both forward and backward language models using attention mechanism when zeta (ζ) is inactive and weight dropout parameters varied by a factor of DM

Dataset	Type of model	DM	Train loss	Valid loss	Accuracy	F1 score
Twitter/ **RLDD**/ *IMDb*	Forward	0.01	0.377/ **0.118**/ *0.021*	0.533/ **0.782**/ *0.235*	0.798/ **0.560**/ *0.938*	0.735/ **0.47**/ *0.938*
		0.5	0.400/ **0.211**/ *0.125*	0.532/ **0.672**/ *0.151*	0.791/ **0.680**/ *0.945*	0.726/ **0.675**/ *0.945*
		0.7	0.507/ **0.374**/ *0.153*	0.514/ **0.591**/ *0.173*	0.801/ **0.720**/ *0.938*	0.737/ **0.712**/ *0.938*
	Backward	0.01	0.383/ **0.239**/ *0.025*	0.551/ **0.847**/ *0.225*	0.790/ **0.520**/ *0.936*	0.724/ **0.342**/ *0.936*
		0.5	0.486/ **0.483**/ *0.129*	0.532/ **0.658**/ *0.150*	0.791/ **0.600**/ *0.945*	0.726/ **0.504**/ *0.945*
		0.7	0.499/ **0.538**/ *0.166*	0.517/ **0.660**/ *0.169*	0.797/ **0.600**/ *0.939*	0.740/ **0.583**/ *0.939*

Design metrics follow the same formatting as mentioned in Table 2

Table 4 Classification results of both forward and backward language models by employing attention mechanism when zeta (ζ) is active and weight dropout parameters remain independent of DM

Dataset	Type of model	Zeta(ζ)	Train loss	Valid loss	Accuracy	F1 score
Twitter/ **RLDD/** *IMDb*	Forward	0.01	0.588/ **0.207/** *0.023*	0.480/ **0.682/** *0.270*	0.812/ **0.560/** *0.933*	0.684/ **0.549/** *0.933*
		0.5	0.400/ **0.245/** *0.015*	0.490/ **0.597/** *0.170*	0.798/ **0.880/** *0.940*	0.740/ **0.880/** *0.940*
		0.7	0.363/ **0.255/** *0.017*	0.538/ **0.625/** *0.262*	0.799/ **0.680/** *0.934*	0.733/ **0.667/** *0.934*
	Backward	0.01	0.461/ **0.132/** *0.015*	0.483/ **0.662/** *0.266*	0.808/ **0.520/** *0.930*	0.730/ **0.480/** *0.930*
		0.5	0.472/ **0.367/** *0.018*	0.500/ **0.700/** *0.242*	0.806/ **0.560/** *0.937*	0.745/ **0.430/** *0.937*
		0.7	0.561/ **0.323/** *0.017*	0.518/ **0.671/** *0.247*	0.799/ **0.560/** *0.937*	0.729/ **0.550/** *0.937*

This table follows the same formatting rules as followed in Table 2

Next, we present the results of the proposed model. These experiments have been performed using aforementioned variations of DM and *zeta*. In Table 6, the scaling factor ζ is not active. We have varied the weight dropout parameters by the factor of (DM) lying in the range [0,1], for both Forward and Backward Language models independently. In Table 7, scaling factor ζ becomes active but DM goes naught. Both ζ and DM become active in Table 8. Both are varied according to each other in their own limits. We have chosen the values of ζ and DM in such a way that their effect on the model can be tested in their entire range efficiently.

Following this, we have performed a comparison of both aggregate architectures, namely concat pool and attention. This comparison is shown in Table 9. The final ensemble classification accuracy of both Forward and Backward Language models is the measure of comparison for all the databases. Table 10 shows how effectively our model has performed when juxtaposed to other models. Along with ensemble classification accuracy results, we have illustrated graphical results in Figs. 6, 7 and 8 relating Loss and Learning Rate for both forward and backward language models, respectively.

We have used four Key Performance Indicators to evaluate the effectiveness of our approach: Accuracy, Precision, Recall and F1 score. Although widely used, accuracy alone is not enough information to decide whether the proposed model makes robust predictions or not. The reason behind this is the overwhelming nature of unbalanced databases, where high accuracy can be achieved by an over- or under-fit model which do not generalize well for new data. Therefore, in order to fully evaluate the model, more performance metrics are required. Those metrics are Precision, Recall and F1 score. We have also used the three aforementioned performance metrics to test the behavior of our model.

1. Accuracy refers to the overall correctness of classification. It measures the ratio of correctly classified instances over the total number of instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where

- *True Positive (TP)* Prediction is positive and it is true.
- *True Negative (TN)* Prediction is negative and it is true.
- *False Positive (FP)* Prediction is positive and it is false.
- *False Negative (FN)* Prediction is negative and it is false.

2. Precision is the ratio of the correctly positive labeled instances to all the positive labeled instances. In other words, it defines the reliability of results given by the model.

$$Precision = \frac{TP}{TP + FP}$$

3. Recall tells how many of the actual positive cases are predicted by the model correctly. It expresses how well the model is able to detect a particular instance.

Table 5 Classification results of both forward and backward language models by using attention mechanism when zeta (ζ) is varied according to certain DM values

Dataset	Type of model	DM	Zeta (ζ)	Train loss	Valid loss	Accuracy	F1 score
Twitter/ **RLDD**/ *IMDb*	Forward	0.01	0.01	0.267/ **0.133**/ *0.015*	0.560/ **0.621**/ *0.231*	0.797/ **0.680**/ *0.938*	0.737/ **0.679**/ *0.938*
		0.01	0.5	0.597/ **0.344**/ *0.022*	0.506/ **0.653**/ *0.250*	0.806/ **0.600**/ *0.937*	0.730/ **.594**/ *0.937*
		0.01	0.7	0.365/ **0.418**/ *0.011*	0.479/ **0.690**/ *0.248*	0.823/ **0.600**/ *0.938*	0.768/ **0.583**/ *0.938*
		0.5	0.01	0.439/ **0.360**/ *0.116*	0.513/ **0.782**/ *0.177*	0.803/ **0.520**/ *0.932*	0.723/ **0.404**/ *0.932*
		0.5	0.5	0.489/ **0.221**/ *0.115*	0.487/ **0.603**/ *0.161*	0.802/ **0.680**/ *0.939*	0.742/ **0.652**/ *0.939*
		0.5	0.7	0.474/ **0.381**/ *0.105*	0.487/ **0.675**/ *0.160*	0.812/ **0.600**/ *0.943*	0.746/ **0.565**/ *0.943*
	Forward	0.7	0.01	0.473/ **0.541**/ *0.168*	0.490/ **0.641**/ *0.169*	0.807/ **0.720**/ *0.938*	0.742/ **0.712**/ *0.938*
		0.7	0.5	0.510/ **0.478**/ *0.154*	0.485/ **0.643**/ *0.169*	0.807/ **0.720**/ *0.936*	0.738/ **0.712**/ *0.936*
		0.7	0.7	0.505/ **0.333**/ *0.153*	0.481/ **0.633**/ *0.168*	0.813/ **0.720**/ *0.937*	0.749/ **0.688**/ *0.937*
Twitter/ **RLDD**/ *IMDb*	Backward	0.01	0.01	0.266/ **0.200**/ *0.017*	0.515/ **0.731**/ *0.276*	0.802/ **0.680**/ *0.929*	0.736/ **0.666**/ *0.929*
		0.01	0.5	0.484/ **0.253**/ *0.025*	0.498/ **0.719**/ *0.227*	0.807/ **0.600**/ *0.940*	0.738/ **0.594**/ *0.940*
		0.01	0.7	0.407/ **0.187**/ *0.024*	0.487/ **0.707**/ *0.235*	0.810/ **0.600**/ *0.937*	0.754/ **0.594**/ *0.937*
		0.5	0.01	0.475/ **0.488**/ *0.130*	0.541/ **0.725**/ *0.172*	0.795/ **0.520**/ *0.937*	0.731/ **0.448**/ *0.937*
		0.5	0.5	0.503/ **0.522**/ *0.117*	0.479/ **0.662**/ *0.159*	0.805/ **0.560**/ *0.943*	0.740/ **0.548**/ *0.943*
		0.5	0.7	0.402/ **0.595**/ *0.119*	0.509/ **0.617**/ *0.159*	0.795/ **0.680**/ *0.942*	0.734/ **0.404**/ *0.942*
		0.7	0.01	0.511/ **0.437**/ *0.171*	0.483/ **0.684**/ *0.176*	0.813/ **0.520**/ *0.934*	0.749/ **0.519**/ *0.934*
		0.7	0.5	0.559/ **0.618**/ *0.153*	0.492/ **0.649**/ *0.171*	0.808/ **0.680**/ *0.938*	0.751/ **0.666**/ *0.938*
		0.7	0.7	0.500/ **0.565**/ *0.137*	0.499/ **0.682**/ *0.165*	0.807/ **0.600**/ *0.941*	0.738/ **0.540**/ *0.941*

This table follows the same formatting rules as mentioned in Table 2

Table 6 Classification results of both forward and backward language models of our proposed model when zeta (ζ) is inactive and weight dropout parameters varied by a factor of DM

Dataset	Type of model	DM	Train loss	Valid loss	Accuracy	F1 score
Twitter/ **RLDD**/ *IMDb*	Forward	0.01	0.420/ **0.165**/ *0.013*	0.585/ **0.562**/ *0.245*	0.782/ **0.680**/ *0.933*	0.718/ **0.679**/ *0.933*
		0.5	0.473/ **0.231**/ *0.118*	0.495/ **0.583**/ *0.172*	0.805/ **0.760**/ *0.940*	0.736/ **0.760**/ *0.940*
		0.7	0.507/ **0.319**/ *0.156*	0.485/ **0.630**/ *0.174*	0.819/ **0.720**/ *0.938*	0.752/ **0.713**/ *0.938*
	Backward	0.01	0.472/ **0.227**/ *0.019*	0.522/ **0.552**/ *0.247*	0.797/ **0.640**/ *0.938*	0.722/ **0.618**/ *0.938*
		0.5	0.503/ **0.416**/ *0.122*	0.491/ **0.547**/ *0.171*	0.807/ **0.760**/ *0.939*	0.736/ **0.750**/ *0.939*
		0.7	0.548/ **0.498**/ *0.172*	0.486/ **0.620**/ *0.169*	0.815/ **0.600**/ *0.937*	0.744/ **0.583**/ *0.937*

Design metrics follow the same formatting as mentioned in Table 2

Table 7 Classification results of both forward and backward language models of our proposed model when zeta (ζ) is active and weight dropout parameters remain independent of DM

Dataset	Type of model	Zeta (ζ)	Train loss	Valid loss	Accuracy	F1 score
Twitter/ **RLDD**/ *IMDb*	Forward	0.01	0.602/ **0.179**/ *0.026*	0.525/ **0.548**/ *0.273*	0.804/ **0.760**/ *0.930*	0.726/ **0.756**/ *0.93*
		0.5	0.410/ **0.116**/ *0.025*	0.494/ **0.580**/ *0.270*	0.798/ **0.720**/ *0.932*	0.746/ **0.713**/ *0.932*
		0.7	0.596/ **0.155**/ *0.014*	0.517/ **0.557**/ *0.259*	0.808/ **0.680**/ *0.936*	0.735/ **0.679**/ *0.936*
	Backward	0.01	0.462/ **0.152**/ *0.027*	0.484/ **0.707**/ *0.286*	0.809/ **0.600**/ *0.922*	0.734/ **0.540**/ *0.922*
		0.5	0.465/ **0.226**/ *0.023*	0.488/ **0.762**/ *0.247*	0.811/ **0.600**/ *0.932*	0.738/ **0.504**/ *0.932*
		0.7	0.460/ **0.219**/ *0.022*	0.491/ **0.634**/ *0.253*	0.806/ **0.600**/ *0.931*	0.728/ **0.594**/ *0.931*

This table follows the same formatting rules as followed in Table 2

$$Recall = \frac{TP}{TP + FN}$$

4. F1 Score is the harmonic mean of precision and recall. The F1 Score mentioned in our results is "macro" averaged because it lets all the labels contribute equally regardless of their appearance in the corpus.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Hyperparameters

The model hyperparameters are beneficial as their values are used to control the learning process. Our model consists of embedding size of 400 units, 3 layers with 1152 units per layer. It has been trained with Back Propagation Through Time (BPTT) technique with a batch size of 70. For classification, we have used dropout of 0.4, 0.05 for input and output embedding layers, respectively. Adam optimization method is used with parameter values $\beta_1 = 0.9$, $\beta_2 = 0.99$.

Table 8 Classification results of both forward and backward language models of our proposed model when zeta (ζ) is varied according to certain DM values

Dataset	Type of model	DM	Zeta (ζ)	Train loss	Valid loss	Accuracy	F1 score
Twitter/ **RLDD**/ *IMDb*	Forward	0.01	0.01	0.596/ **0.222**/ *0.023*	0.524/ **0.544**/ *0.239*	0.801/ **0.760**/ *0.933*	0.717/ **0.750**/ *0.933*
		0.01	0.5	0.596/ **0.161**/ *0.026*	0.505/ **0.540**/ *0.254*	0.807/ **0.720**/ *0.933*	0.729/0 **.713**/ *0.933*
		0.01	0.7	0.583/ **0.179**/ *0.016*	0.519/ **0.666**/ *0.253*	0.802/ **0.680**/ *0.933*	0.724/ **0.675**/ *0.933*
		0.5	0.01	0.497/ **0.256**/ *0.117*	0.501/ **0.604**/ *0.178*	0.809/ **0.720**/ *0.933*	0.750/ **0.713**/ *0.933*
		0.5	0.5	0.497/ **0.230**/ *0.111*	0.495/ **0.599**/ *0.157*	0.811/ **0.680**/ *0.943*	0.750/ **0.675**/ *0.943*
		0.5	0.7	0.479/ **0.457**/ *0.106*	0.485/ **0.553**/ *0.161*	0.814/ **0.760**/ *0.942*	0.749/ **0.760**/ *0.942*
	Forward	0.7	0.01	0.511/ **0.452**/ *0.175*	0.492/ **0.604**/ *0.176*	0.814/ **0.720**/ *0.931*	0.743/ **0.720**/ *0.931*
		0.7	0.5	0.517/ **0.333**/ *0.151*	0.490/ **0.615**/ *0.166*	0.802/ **0.640**/ *0.939*	0.731/ **0.638**/ *0.939*
		0.7	0.7	0.511/ **0.295**/ *0.153*	0.491/ **0.662**/ *0.168*	0.813/ **0.560**/ *0.937*	0.744/ **0.533**/ *0.937*
Twitter/ **RLDD**/ *IMDb*	Backward	0.01	0.01	0.365/ **0.288**/ *0.025*	0.535/ **0.568**/ *0.283*	0.808/ **0.680**/ *0.922*	0.748/ **0.632**/ *0.922*
		0.01	0.5	0.464/ **0.239**/ *0.030*	0.491/ **0.639**/ *0.232*	0.811/ **0.640**/ *0.935*	0.732/ **0.618**/ *0.935*
		0.01	0.7	0.370/ **0.346**/ *0.028*	0.515/ **0.596**/ *0.239*	0.809/ **0.680**/ *0.933*	0.749/ **0.603**/ *0.933*
		0.5	0.01	0.548/ **0.595**/ *0.132*	0.493/ **0.633**/ *0.174*	0.811/ **0.760**/ *0.935*	0.732/ **0.750**/ *0.935*
		0.5	0.5	0.507/ **0.357**/ *0.121*	0.483/ **0.629**/ *0.163*	0.809/ **0.720**/ *0.939*	0.744/ **0.713**/ *0.939*
		0.5	0.7	0.501/ **0.322**/ *0.121*	0.487/ **0.679**/ *0.161*	0.811/ **0.560**/ *0.940*	0.749/ **0.548**/ *0.940*
		0.7	0.01	0.53/ **0.568**/ *0.176*	0.483/ **0.638**/ *0.181*	0.813/ **0.760**/ *0.929*	0.749/ **0.750**/ *0.929*
		0.7	0.5	0.551/ **0.456**/ *0.153*	0.484/ **0.662**/ *0.171*	0.813/ **0.560**/ *0.938*	0.743/ **0.533**/ *0.938*
		0.7	0.7	0.557/ **0.400**/ *0.144*	0.495/ **0.559**/ *0.172*	0.811/ **0.720**/ *0.934*	0.731/ **0.703**/ *0.934*

This table follows the same formatting rules as mentioned in Table 2

Table 9 Comparison of experimental results of ensemble classification accuracy of Concat Pool as used in our proposed model to existing state-of-the-art Attention mechanism with respect to same variations of DM and zeta as used in aforementioned experimental results for all the three databases

Parameter (DM/zeta)	Ensemble accuracy (concat pool/attention)		
	RLDD	Twitter	IMDB
0.01/1.0	70/52	80.05/80.05	94.68/94.79
0.5/1.0	76/52	80.33/79.64	94.59/94.28
0.7/1.0	72/64	81.18/80.23	94.35/94.24
0.0/0.01	72/60	81.97/81.02	93.53/94.14
0.0/0.5	68/76	82.27/81.52	94.31/94.64
0.0/0.7	72/64	81.66/81.50	94.21/94.51
0.01/0.01	76/60	80.94/81.15	93.89/94.30
0.01/0.5	74/70	81.01/82.10	94.58/94.64
0.01/0.7	84/58	80.36/79.97	94.63/94.23
0.5/0.01	64/52	81.43/80.91	94.14/93.99
0.5/0.5	78/66	81.40/81.28	94.80/94.77
0.5/0.7	72/72	81.93/81.22	94.94/94.74
0.7/0.01	80/66	82.21/80.70	94.68/94.54
0.7/0.5	70/80	81.25/81.56	94.25/94.23
0.7/0.7	76/74	81.76/81.52	94.29/93.79

The three sections divided in this table are in accordance with the same three variations using DM and zeta

We have used a batch size of 32, a learning rate of 0.004 and 0.01 for fine-tuning the language model and classifier, respectively. Weight dropout parameter and scaling factor (ζ) have been varied in their respective ranges. We have used the concept of bidirectional modeling by fine-tuning forward and backward language models separately and ensembling their predictions at the end.

6.1 Comprehensive comparison

In this section, we present comparison of our evaluation results with the existing state-of-the-art models on sentiment analysis task. First 2 approaches namely Logistic Regression and Linear SVM are traditional machine learning techniques as mentioned in Sect. 2. Next 2 approaches namely GloVe+biLSTM and fastText are non-deep learning methods for sentiment analysis. These are also known as distributed representations of words. Rest of the strategies including ours are deep learning models. Out of these strategies, all are based on transfer learning except for ANN. Apart from the top 4 approaches, every approach is capable of automatically extracting features from input data without the help of feature engineering.

Table 10 shows that classification accuracy of our model is preferable when tested on three conventional datasets, as mentioned in Sect. 3. Considering the algorithm effectiveness, fastText has proved salient among other non-deep learning techniques in its category. Our proposed model, on the other hand, has achieved superior results not only among the deep learning approaches but all the eight approaches, as mentioned above. The classification accuracy of our model is 82.27%, 84% and 94.94% for US airline twitter, Real life deception detection and IMDb datasets respectively. In contrast to ULMFit, we planned to bring bidirectional contextualization using ensemble methods. We also focused on regularization by generalizing the DropConnect method. Our aim is to create an ambitious yet reasonable framework for sentiment analysis task. Overall, our model has performed remarkably on experimental datasets and with such performance, we can consider it as a robust framework for sentiment classification problems.

All the variations of our model have been explained independently in Tables 6, 7 and 8 as shown above. The

Table 10 Comparison of experimental results of classification accuracy of our proposed model to existing state-of-the-art models

Model	Percentage accuracy on various datasets		
	RLDD (%)	Twitter (%)	IMDb (%)
Logistic Regression (Krishnamurthy et al. 2018)	64.57	64.31	52
Linear SVM (Krishnamurthy et al. 2018)	56.43	64	52
Glove+biLSTM (Abid et al. 2020)	58.33	69.08	87.9
fastText (Abid et al. 2020)	66.67	74.16	88.66
ANN (Shaukat et al. 2020)	80.16	79.4	91
ULMFit (Howard and Ruder 2018)	62.5	80.6	93.95
BERT-base (Devlin et al. 2019)	76.92	81.57	93.2
Soft Attention (Bahdanau et al. 2015)	80	82.10	94.63
Our model	**84**	**82.27**	**94.94**

Highest accuracy values are highlighted in **bold**

combined results of all model variations are illustrated in Figs. 9 and 10. The two bar graphs presented here show ensemble classification accuracy of both forward and backward language models for all experimental datasets. Sensitivity analysis of our model variations are explained using two parameters: DM and zeta(ζ). In first bar graph, every value of DM is varied for all zeta values in the entire range. Correspondingly, second graph illustrates variation of every value of zeta with DM.

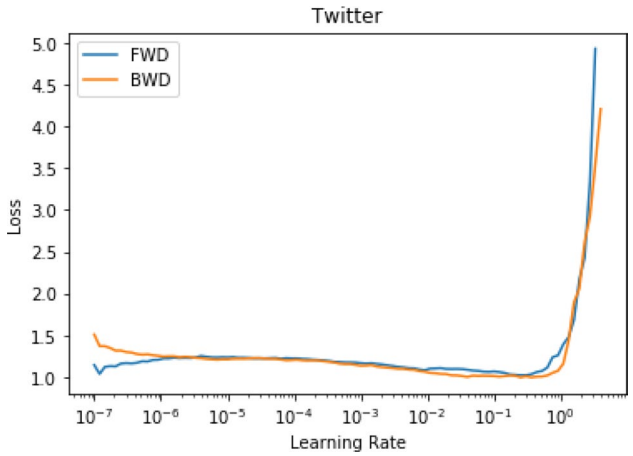

Fig. 6 Loss versus learning rate of Twitter data

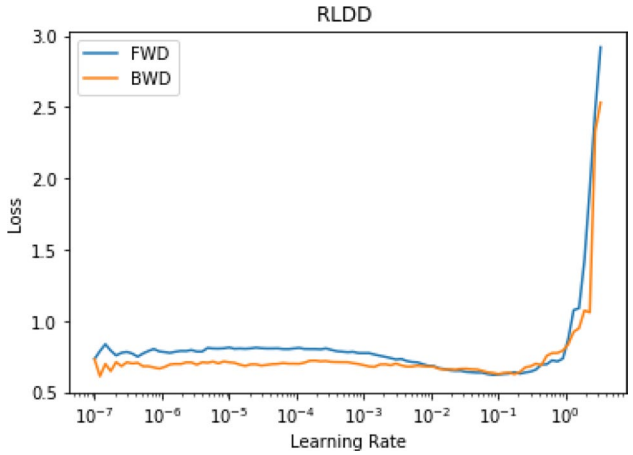

Fig. 7 Loss versus learning rate of RLDD data

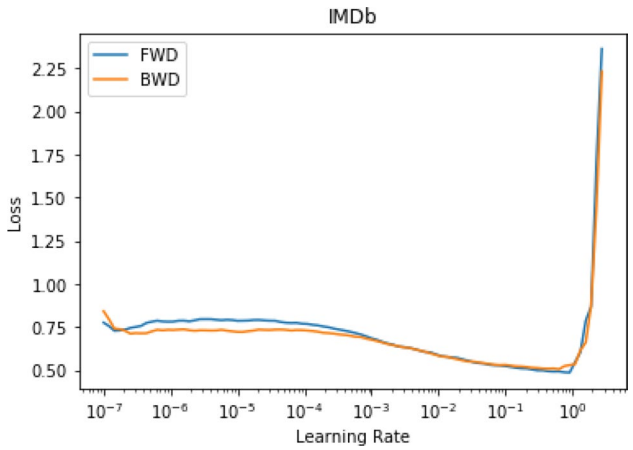

Fig. 8 Loss versus learning rate of IMDb data

For IMDb, the accuracy is at its best when around 50% of dropout parameters and 70% of recurrent weights of AWD-LSTM are scaled. DM is relatively less sensitive for this case. In case of RLDD, our model achieves maximum accuracy when only 1% of dropout parameters and 70% of AWD-LSTM weights are scaled. Thus, our model is most sensitive to zeta for this corpus. For another counterpart Twitter, our model works best when we scale 50% of the AWD-LSTM weights. Clearly it does not depend on the dropout parameter scaling and is sensitive to weights scaling only. Altogether, it can be observed that our model is relatively more sensitive to weight scaling factor zeta (ζ). Keeping DM into consideration, we can say that performance of our model varies directly with AWD-LSTM scaled weights in their respective limits.

Overall, the combination of DM = 0.5 and zeta = 0.7 proves best for IMDb where our model achieves highest classification accuracy of 94.94%. Similarly, in case of RLDD dataset, it achieves highest accuracy of 84% when DM = 0.01 and zeta = 0.7. On Twitter, our model attains maximum accuracy of 82.27% for DM = 0.0 and zeta = 0.5.

7 Conclusion and future work

In this manuscript, we have focused on sentiment classification task and proposed a new transfer learning based model. It intends to work on challenging sentiment analysis tasks such as contextualization and regularization. The main steps followed by our model include: (1) pre-train a language model (2) fine-tune the language model on target data by scaling recurrent weights of AWD-LSTM using a scaling factor zeta(ζ). To introduce bidirectionality, we have performed both forward and backward language modeling based on state-of-the-art transfer learning based methods. Our model is an ensemble representation of forward and backward language models.

After an extensive and thorough research, we have demonstrated that the model variations, ζ+concat pool and ζ +attention perform extremely well as compared to their conventional versions where no zeta is incorporated. The performance of the model is intensified when ζ is included suitably in the model. In addition to this, our model has surpassed the experimental results of soft attention mechanism too. Hitherto, the comparison with attention layer has been propounded especially on the grounds that both attention and concat pool are state-of-the-art feature aggregate

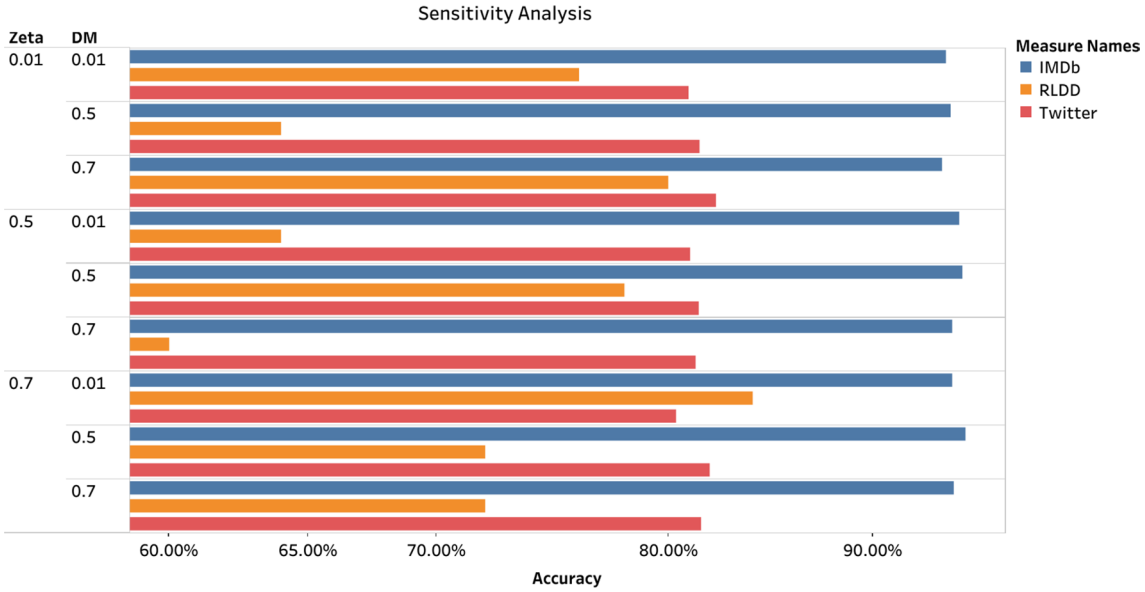

Fig. 9 Sensitivity analysis of classification accuracy w.r.t zeta (ζ) for various experimental datasets

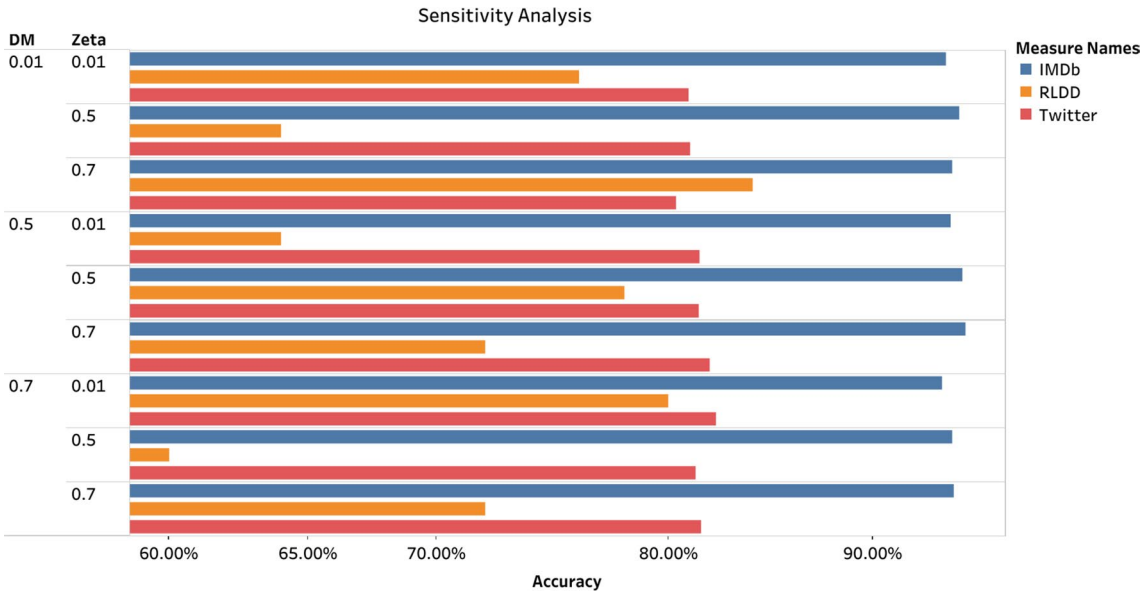

Fig. 10 Sensitivity analysis of classification accuracy w.r.t DM for various experimental datasets

architectures. That is why separate research has been carried out using both architectures individually and our model has emerged as the superior one.

Finally, it has been validated that the incorporation of ζ and its variation with DM has yielded superior results. This is so because with the companionship of these parameters, our model is no longer reckoning on just maximum and average values but has the hold of other dominant features of the embedding layer. The intention is to learn significant features of embeddings rather than just presuming that concatenation of maximum and average values or just attention mechanism can provide finest results. Therefore, we have arrived at a conclusion that inclusion of zeta in our model makes an apt and excellent substitute for attention mechanism. Our model has performed remarkably well in terms of classification accuracy when compared to other state-of-the-art approaches.

As mentioned, we have evaluated the efficiency of our model on three widely-used databases. All these databases, as mentioned in Table 1, have a diversified range. We have obtained outstanding experimental results for our model compared to the ones obtained from other state-of-the-art frameworks. Thus, it shows that our proposed model is versatile and robust in nature.

In future, we plan to extend our work for multi-class text classification by incorporating Transformer based models with some other attention mechanisms.

Acknowledgements This work was supported by free academic credits from Google Cloud Platform.

References

Abid F, Li C, Alam M (2020) Multi-source social media data sentiment analysis using bidirectional recurrent convolutional neural networks. Comput Commun 157:102–115

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. ICLR, San Diego

Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155

Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc., Newton

Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146

Bouazizi M, Ohtsuki T (2017) A pattern-based approach for multi-class sentiment analysis in twitter. IEEE Access 5:20617–20639

Cambria E (2016) Affective computing and sentiment analysis. IEEE Intell Syst 31:102–107

Cambria E, Poria S, Gelbukh A, Thelwall M (2017) Sentiment analysis is a big suitcase. IEEE Intell Syst 32:74–80

Chen T, Xu R, He Y, Wang X (2017) Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. Expert Syst Appl 72:221–230

Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537

Crowdflower (2016) Airline Twitter Sentiment. https://data.world/crowdflower/airline-twitter-sentiment. Online accessed 01 December 2019

de Araujo PHL, de Campos TE, de Sousa MMS (2020) Inferring the source of official texts: can SVM beat ULMFiT?. Springer, Evora, pp 76–86

Devlin J, Chang M-W, Lee K, Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding, NAACL-HLT, Association for Computational Linguistics

Glorot X, Bordes A, Bengio Y (2011) Domain adaptation for large-scale sentiment classification: a deep learning approach. ICML, Bellevue, pp 513–520

Gupta C, Jain A, Joshi N (2019) A novel approach to feature hierarchy in aspect based sentiment analysis using OWA operator. Springer, Chandigarh, pp 661–667

Haddoud M, Mokhtari A, Lecroq T, Abdeddaïm S (2016) Combining supervised term-weighting metrics for SVM text classification with extended term representation. Knowl Inf Syst 3:909–931

Hermann KM, Blunsom P (2013) The role of syntax in vector space models of compositional semantics. Association for Computational Linguistics, Sofia, pp 894–904

Howard J, Ruder S (2018) Universal language model fine-tuning for text classification. Assoc Comput Linguist 1:328–339

Jean-François P (2017) Feature engineering for deep learning. https://medium.com/inside-machine-learning/feature-engineering-for-deep-learning-2b1fc7605ace. Online accessed 15 December 2019

Jiang M, Liang Y, Feng X, Fan X, Pei Z, Xue Yu, Guan R (2018) Text classification based on deep belief network and softmax regression. Neural Comput Appl 29:61–70

Jianqiang Z, Xiaolin G (2017) Comparison research on text preprocessing methods on twitter sentiment analysis. IEEE Access 5:2870–2879

Joulin A, Grave E, Bojanowski P, Mikolov T (2017) Bag of tricks for efficient text classification. Association for Computational Linguistics, Valencia, pp 427–431

Krishnamurthy G, Majumder N, Poria S, Cambria E (2018) A deep learning approach for multimodal deception detection. arXiv preprint arXiv:1803.00344

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of machine learning research, pp 1188–1196, Beijing

Liu R, Shi Y, Ji C, Jia M (2019) A survey of sentiment analysis based on transfer learning. IEEE Access 7:85401–85412

Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis, Portland, Association for Computational Linguistics, pp 142–150

Manning CD, Manning CD, Schütze H (1999) Foundations of statistical natural language processing. MIT press, Cambridge

McCann B, Bradbury J, Xiong C, Socher R (2017) Learned in translation: contextualized word vectors. Association for Computing Machinery, California, pp 6294–6305

Merity S, Keskar NS, Socher R (2018) Regularizing and optimizing LSTM language models. ICLR

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013a) Distributed representations of words and phrases and their compositionality. Association for Computing Machinery, New York, pp 3111–3119

Mikolov T, Chen K, Corrado G, Dean J (2013b) Efficient estimation of word representations in vector space. In: Proceedings of workshop at ICLR

Mirończuk MM, Protasiewicz J (2018) A recent overview of the state-of-the-art elements of text classification. Expert Syst Appl 106:36–54

Neelakantan A, Shankar J, Passos A, McCallum A (2015) Efficient non-parametric estimation of multiple embeddings per word in vector space. Association for Computational Linguistics, Doha, pp 1059–1069

Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans knowl Data Eng 22:1345–1359

Pathak AR, Agarwal B, Pandey M, Rautaray S (2020) Application of deep learning approaches for sentiment analysis. Springer, Singapore, pp 1–31

Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. Association for Computational Linguistics, Qatar, Valencia, pp 1532–1543

Pérez-Rosas V, Abouelenien M, Mihalcea R, Burzo M (2015) Deception detection using real-life trial data. Association for Computing Machinery, Seattle, pp 59–66

Peters ME, Ammar W, Bhagavatula C, Power R (2017) Semi-supervised sequence tagging with bidirectional language models. Association for Computational Linguistics, Vancouver, pp 1756–1765

Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of NAACL-HLT, pp 2227–2237

Rane A, Kumar A (2018) Sentiment classification system of twitter data for US airline service analysis. IEEE, Tokyo, pp 769–773

Saif H, He Y, Fernandez M, Alani H (2016) Contextual semantics for sentiment analysis of Twitter. Inf Process Manag 52:5–19

Shaukat Z, Zulfiqar AA, Xiao C, Azeem M, Mahmood T (2020) Sentiment analysis on IMDB using lexicon and neural networks. SN Appl Sci 2:1–10

Soleymani M, Garcia D, Jou B, Schuller B, Chang S-F, Pantic M (2017) A survey of multimodal sentiment analysis. Image Vis Comput 65:3–14

Turney PD, Pantel P (2010) From frequency to meaning: vector space models of semantics. J Artif Intell Res 37:141–188

Vaswani A, Shazeer N, Parmar N, Jakob U, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008

Wang Y, Hou Y, Che W, Liu T (2020) From static to dynamic word representations: a survey. Int J Mach Learn Cybern 11:1611–1630

Wu Y, Li J, Wu J, Chang J (2020) Siamese capsule networks with global and local features for text classification. Neurocomputing 390:88–98

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhutdinov R, Zemel R, Bengio Y (2015) Show, attend and tell: neural image caption generation with visual attention. In: Proceedings of the 32nd international conference on machine learning, PMLR, vol 37, pp 2048–2057

Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks?. Association for Computing Machinery, Montreal, pp 3320–3328

Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13:55–75

Zheng J, Cai F, Chen H, de Rijke M (2020) Pre-train, Interact, Fine-tune: a novel interaction representation for text classification. Inf Process Manag 57:102215