Working in Detail: How LSTM Hyperparameter Selection Influences Sentiment Analysis Results

Nicholas Daniel Derra and Daniel Baier

Abstract Sentiment analysis of written customer reviews is a powerful way to generate knowledge about customer attitudes for future marketing activities. Meanwhile, Deep Learning as the most powerful machine learning method is of particular importance for sentiment analysis tasks. Due to this current relevance, an LSTM network based on a literature review to solve the challenging classification task of the IMDB Large Movie Dataset is created. Hyperparameters are varied separately from each other to better understand their single influences on the overall model accuracy. Furthermore, we transformed variants with positive impacts into a final model in order to investigate whether the impacts can be cumulated. While preparing the amount of training data and the number of iteration steps resulted in a higher accuracy, pre-trained word vectors and higher network capacity did not work well separately. Even though implementing the variants with positive influences together raised the model´s performance, the improvement was lower than some single variants.

Nicholas Daniel Derra
BF/M-Bayreuth, Mainstraße 5, 95444 Bayreuth, Germany,
✉ nicholas.derra@bfm-bayreuth.de

Daniel Baier
University of Bayreuth, Chair of Marketing and Innovation, Universitätsstraße 30, 95447 Bayreuth, Germany,
✉ daniel.baier@uni-bayreuth.de

Archives of Data Science, Series A
(Online First)
KIT Scientific Publishing
Vol. 6, No. 1, 2020

DOI: 10.5445/KSP/1000098011/10
ISSN 2363-9881

1 Introduction

Sentiment analysis (SA) has been one of the largest fields of research in natural language processing (NLP), data mining, text mining and information retrieval since the beginning of the 21st century. Due to the ever-increasing use of internet and online activities (e-commerce, forums, blogs and social networks) for presenting personal opinions about products and services, the analysis of the resulting huge amounts of data (Big Data) is of particular importance for marketing managers (Zhang et al, 2018b). Meanwhile, Deep Learning (DL) algorithms deliver stronger results in processing sequential text data for SA tasks than other Machine Learning (ML) methods do (LeCun et al, 2015). For this, the current literature focuses on the development of models that classify popular benchmark datasets (IMDB Large Movie Dataset by Maas et al (2011); Yelp Dataset by Zhang et al (2015)) with a new accuracy highscore. We argue that in this context only the overall performance of an architecture is observed while the various influences of individual hyperparameters on the model performance are insufficiently analysed. For this reason, the separate effects of various hyperparameters within an LSTM network for the IMDB Large Movie Dataset sentiment analysis task are observed through separate variation. Simultaneously, after a short introduction (Section 1), the discussion of theoretical backgrounds including SA (Section 2.1) and DL models for SA (Section 2.2) as well as the description of the IMDB dataset (Section 3.1) and related work (Section 3.2), an LSTM which is able to solve the IMDB SA task with high accuracy is constructed (Section 3.3). Within this model, 8 hyperparameters are separately varied to investigate their impact on classification performance. Subsequently, the variations with a positive impact on validation accuracy are transformed into a final model in order to cumulate the effects. This final model is then compared with the single hyperparameter variants by test accuracy. In addition, the machine times required are also measured (Section 3.4). Finally, the results are discussed in Section 4.

2 Theoretical Background

2.1 Sentiment Analysis

SA, also called mood analysis, is the field of computational studies of emotions as well as opinions, feelings, evaluations and attitudes towards objects such as products, services, organizations, individuals, events, topics and issues as well as their characteristics (Ain et al, 2017; Medhat et al, 2014). They are analysed in forums, blogs, social networks, e-commerce websites, reports and other internet sources (Ravi and Ravi, 2015). SA is a subset of both NLP and affective computing (AC) (Yadollahi et al, 2017; Zhang et al, 2018a) and can therefore be seen as an intersection of both areas of research. It is carried out by methods of information retrieval and data mining (Ravi and Ravi, 2015). While the different SA tasks can be correctly subdivided into the subareas of opinion mining (analysis of contained opinions in texts) and emotion mining (analysis of contained emotions in texts) (Yadollahi et al, 2017), a more comprehensive approach summarizing opinions and emotions (Ravi and Ravi, 2015) seems to be more effective. Since the concept of a sentiment encompasses both opinions and emotions, a precise SA can only be achieved by analysing both areas simultaneously (Ain et al, 2017; Medhat et al, 2014).

While SA is often used as a synonym for sentiment or polarity classification, it is considered to be the central SA task (Cambria et al, 2013). However, in this article this trend of literature is taken into account (see inter alias Araque et al (2017) and Medhat et al (2014)) so the term SA is used interchangeably after the various fields of SA tasks were shown. A sentiment, respectively polarity classification, is the recognition of the sentiment orientation within a text and the classification into one of at least two classes. As the most common task in SA, the polarity classification classifies texts according to their opinion into a predefined sentiment polarity, whereby both binary, tertiary and finer n-grade classifications are possible (Ravi and Ravi, 2015). Polarity classification can take place on three granularity levels, regardless of the classification object (opinion, emotion or both). For this, the document, sentence and aspect level are differentiated (Medhat et al, 2014; Yadollahi et al, 2017; Zhang et al, 2018a) where the polarity classification at the document level is considered to be the most common. At the document level, a complete text document is considered as the smallest unit. This document expresses an overall positive or negative

opinion or emotion and it is usually assigned either to the positive or the negative class (Aggarwal and Aggarwal, 2017; Medhat et al, 2014; Yadollahi et al, 2017; Zhang et al, 2018a). Yet, the length of the document is irrelevant (Yadollahi et al, 2017). At this level it is assumed that not every single sentence contains an opinion relating to the subject so the document contains irrelevant sentences (Aggarwal and Aggarwal, 2017). Since the IMDB sentiment classification task is to classify film ratings of different lengths and without focusing on specific aspects with respect to their polarity, the IMDB task is performed at the document level.

Approach	Reference	Accuracy
Support Vector Machines (SVM)	Wang and Manning (2012)	89.16%
Maximum Entropy (ME)	Brychcín and Habernal (2013)	92.24%
Naive Bayes (NB)	Narayanan et al (2013)	88.80%
NB-SVM	Mesnil et al (2014)	91.87%
Decision Trees (DT)	Zhou and Feng (2017)	89.16%
Deep Learning (DL)	Howard and Ruder (2018)	95.40%

Table 1: ML methods for the IMDB sentiment classification task.

The polarity classification approaches can be divided into ML-based, lexicon-based, while hybrid approaches are ultimately a combination of ML with a previously created lexicon (Maynard and Funk, 2011; Ravi and Ravi, 2015). ML techniques treat sentiment classifications as text classification tasks and use syntactic and linguistic properties to solve problems (Medhat et al, 2014). They clearly outperform the semantic approaches in dealing with specific tasks (Ravi and Ravi, 2015). They are divided into methods of supervised, unsupervised and semi-supervised learning, whereby unsupervised ML methods only play a minor role in SA research and are only marginally or not explained in the relevant overview literature (Medhat et al, 2014; Ravi and Ravi, 2015; Yadollahi et al, 2017).

For polarity classification with supervised learning, probabilistic classifiers such as Bayesian Networks, Naive Bayes classifiers and Maximum Entropy classifiers, linear classifiers such as Support Vector Machines and Artificial Neural Networks (Deep Learning), Decision Trees and Rule-based classifiers are frequently used (Medhat et al, 2014; Ravi and Ravi, 2015). In terms of the IMDB

Large Movie Dataset, the classification performance of the different methods is shown in table 1. Thereby, DL models have achieved a large number of correct classification rates higher than 92% in recent years, massively outperforming other ML approaches (compare our literature review in Section 4). The current IMDB benchmark performed with DL achieves 95.40% accuracy (Howard and Ruder, 2018). To summarize, while ML approaches have task-specific higher accuracy than lexicon production, DL outperforms conventional ML.

2.2 Deep Learning

DL approaches are part of the research field of artificial intelligence (AI) (Arel et al, 2010) as well as a methodologically emerging area of ML called Representation Learning. Within DL methods, several stages of representation transformation take place in succession (LeCun et al, 2015). Meanwhile, DL is defined as a class of ML techniques based on Artificial Neural Networks (ANN) that use numerous (hidden) process layers in hierarchical architectures to learn characteristics and recognize patterns from data (Deng, 2011, 2014). However, the depth required for the concept of DL is not uniformly defined in research (Schmidhuber, 2015).

In the context of ANNs, the concept of learning describes a process for updating the network architecture and the weights of neuron connections to efficiently handle a specific task (Jain et al, 1996). In DL, the most commonly used supervised learning algorithm is the backpropagation method for error minimization which allowed to map direct connections of neurons over several layers so that the weights within the ANNs were efficiently learned (Deng, 2014; Schmidhuber, 2015). In general, backpropagation is a special case of the general gradient descent process (Schmidhuber, 2015). This approach by Rumelhart et al (1986) repeatedly adjusts the weights within an ANN to minimize the difference between the actual output vector and the known output vector setpoint for finding an optimal set of weights. The quality of the weights is described by the difference between the actual and target output vectors in a quadratic error function.

Basically, Deep Neural Networks are classified in Feed Forward (FNN) and Recurrent (RNN) Neural Networks (Jain et al, 1996; Schmidhuber, 2015). Furthermore, the forward models are divided into Deep Autoencoders (DAE),

Deep Belief Networks (DBN) and Convolutional Neural Networks (CNN) (Deng, 2014; Zhang et al, 2018b). The recurrent networks were later developed into so-called Long Short Term Memories (LSTM) (Gers et al, 1999; Hochreiter and Schmidhuber, 1997). While DAEs and DBNs are only used for (unsupervised) pre-training in polarity classification tasks, one-dimensional CNNs, but especially RNNs and their powerful relatives LSTMs are able to classify text data very well (LeCun et al, 2015).

RNNs are more powerful than any forward DL model because of their ability to create memories (Schmidhuber, 2015). Due to the backward links, they can account for time sequences and are therefore perfect in processing sequential data, e.g. natural language. RNNs have a cyclic architecture and are able to learn the data properties through a memory from previous inputs (Jain et al, 1996; Zhang et al, 2018a). The memory of an RNN is its ability to process all the elements of a sequence where the input of a unit thus consists of two parts, the current input and the output of previous calculations (Zhang et al, 2018a). This is possible because the information from previous calculations is stored as an internal state within the RNN (LeCun et al, 2015; Zhang et al, 2018a).

However, especially at deep RNNs, the vanishing or exploding gradients during backpropagation training has proved to be very problematic due to long-term dependencies (Bengio et al, 1994; Hochreiter, 1991; Schmidhuber, 2015; Zhang et al, 2018a). To address this phenomenon called fundamental DL problem, LSTMs were developed (Gers et al, 1999; Hochreiter and Schmidhuber, 1997). Today, the most successful RNNs are based on this architecture (Deng, 2014; Schmidhuber, 2015). By using so-called constant error carousels, also known as memory cells, LSTMs are able to remember processes that already took place many time steps ago. These units are connected to themselves with a weight of 1 and thus copy their own state. This connection is linked to another unit, called gate unit, which decides when to erase the learned memory, which information is erased, and which new information is stored in the memory (Gers et al, 1999; Hochreiter and Schmidhuber, 1997; LeCun et al, 2015; Zhang et al, 2018a). Accordingly, a distinction is made between input, forget and output gate units (Hochreiter and Schmidhuber, 1997; Zhang et al, 2018a). The additional possibility of forgetting information and the associated influence on the internal memory enables the effective use of long-term dependencies without vanishing or exploding gradients.

Conventional RNNs and LSTMs can only use the information of previous time steps and therefore do not use all available information of sequential data (Zhang et al, 2018a). For this reason, Bidirectional LSTMs (BiLSTM) have been developed. They consist of two opposing LSTMs stacked on top of each other and are thereby able to process text sequences forward and backward at the same time. Finally, the internal states of both networks are taken into account for calculating the output of the bidirectional network (Schuster and Paliwal, 1997). The bidirectional architecture often provides better sentiment classification results than its unidirectional counterparts since the context between a given word within a text and its subsequent words might be as important as the context to previous words for classifying the sentiment of this word (see, e.g., Howard and Ruder (2018), Johnson and Zhang (2016)).

The danger in supervised learning processes, so-called overfitting, is often caused by a limited amount of training data, too many parameters to be learned (the network capacity) or a large number of training epochs. In such a case, the network learns to identify specific characteristics of the training data which are irrelevant or even obstructive for classifying unknown data (Srivastava et al, 2014). Thus, the task-specific generalization decreases with additional training epochs so the model loses massive usefulness in the analysis of unknown data. RNNs -particularly their bidirectional variants- are quite susceptible to overfitting due to their huge capacity (memory architecture and additional backward neuron connections) so that such models are usually trained with fewer epochs than other architectures in order to learn cumbersome specific features (Hong and Fang, 2015).

In addition, to avoid overfitting, another hyperparameter can be integrated into the model. This method, known as dropout regularization, randomly sets a share of its output per layer to zero, thus extracting a thinned net from the original complex model. The size of this eliminated share is determined by the dropout rate. As a result, the network does not learn any irrelevant patterns contained in the training data which improves unknown data performance a lot (Srivastava et al, 2014). The additional implementation of a recurrent dropout rate makes this method implementable for RNNs (Gal and Ghahramani, 2015).

Since DL algorithms (like other ML methods as well) can not use text data as input, datasets in text form have to be converted into numerical vectors (Zhang et al, 2018a). This results in very high-dimensional property vectors (called One Hot Encoding (OH)) since each word contained must be assigned its own

value. ML applications, therefore, require a feature selection step that removes unimportant properties or words for the task to be performed and thus reduces the dimensionality without reducing the quality of the subsequent classification (Rui et al, 2016; Yang and Pedersen, 1997).

An advantage of sentiment classification via DL is that, in contrast to other ML methods, no feature selection is necessary to avoid these high-dimensional feature vectors since DL models are able to handle high-dimensional data very well and process a feature selection by using the embedding layer for training so-called word embeddings. Using a specific algorithm, it generates smaller numerical vectors and at the same time more information contained by removing the words which are irrelevant for the classification task. Examples of such word embedding algorithms are Word2Vec (Mikolov et al, 2013) and GloVe (Pennington et al, 2014). The word embeddings and the weights are learned simultaneously based on the present training data. If there is insufficient training data for a classification task, pre-trained word embeddings calculated using one of the two algorithms can be used. Such pre-trained vectors are freely available via internet (for Word2Vec: see Google (2013), for GloVe: see Stanford (2014)).

3 Experiments

3.1 Dataset

The IMDB Large Movie Dataset was developed by Maas et al (2011). It was designed to meaningfully test and compare binary sentiment classification methods. This dataset contains 100,000 film ratings from the Internet Movie Database (50,000 labeled and 50,000 unlabeled samples), with each movie represented by a maximum of 30 ratings (Maas et al, 2011). The goal of the IMDB SA task is to correctly classify whether a movie rating is positive or negative. The average length of a review document is 231 words (Wang and Manning, 2012). Within the labeled data, there are 25,000 positive and 25,000 negative reviews each, with only clearly polarized contributions taken into account. Therefore, neutral reviews are not included. The labeled dataset is also divided into 25,000 reviews for training and testing each (Maas et al, 2011). The unlabeled training dataset with 50,000 reviews is intended to, e.g., train a semi-supervised architecture with unsupervised pre-training. This dataset

contains positive, neutral and negative sentiments (Maas et al, 2011). In general, is has to be mentioned that the particular difficulty of classifying film ratings presents a major challenge for all ML methods (Turney, 2002). The basic difficulties and challanges in text analysis, including irony, sarcasm, various word diffractions, synonyms, stop words, etc. are just as demanding as the different lengths of the evaluation documents.

3.2 Related work

Reference	Architecture	Specific Architecture	Test Accuracy
Le and Mikolov (2014)	FNN	PV-FNN	92.58%
Dai and Le (2015)	LSTM	SA-LSTM	92.80%
Johnson and Zhang (2015)	CNN	RE-CNN	93.49%
Dieng et al (2016)	RNN	Topic-RNN	93.72%
Johnson and Zhang (2016)	LSTM	OH-BiLSTM	94.06%
Miyato et al (2016)	LSTM	VA-LSTM	94.09%
Gray et al (2017)	LSTM	Block-Sparse LSTM	94.99%
Radford et al (2017)	LSTM	Byte-Level LSTM	92.88%
Xu et al (2017)	RNN	SSVAE-RNN	92.77%
Howard and Ruder (2018)	LSTM	ULMFiT	95.40%

Table 2: DL models for the IMDB sentiment classification task.

The IMDB Large Movie Dataset classification task has already been solved by a variety of high-performance models, especially during the last 4 years the accuracy of the task has been improved regularly. The currently best architecture was set up by Howard and Ruder (2018) with their ULMFiT model and achieves an accuracy of 95.40% in classifying the IMDB test data. The 10 most powerful DL architectures are listed in Table 2. Within these models, it is noticeable that LSTMs were used disproportionately (6 out of 10). Also, Merity et al (2017) describe these architectures as particularly advantageous for language modelling tasks, as LSTMs are more resistant to the fundamental deep learning problem of the vanishing gradient than other architectures. In addition, Johnson

and Zhang (2015) also demonstrated the efficient use of CNNs for sentiment classification. Although they do not match the accuracy of the best LSTM models, they are convincing due to their competitive classification rates and comparatively low computational effort. However, LSTMs seem to be more promising in setting a new accuracy highscore. The literature review also shows that the implementation of unsupervised elements, especially for pre-training, has positive effects on the performance of deep learning models (8 out of 10 models contained unsupervised learning structures). Nevertheless, due to the question regarding the influences of individual hyperparameters on the overall classification performance, the implementation of unsupervised pre-training is superfluous.

3.3 Model

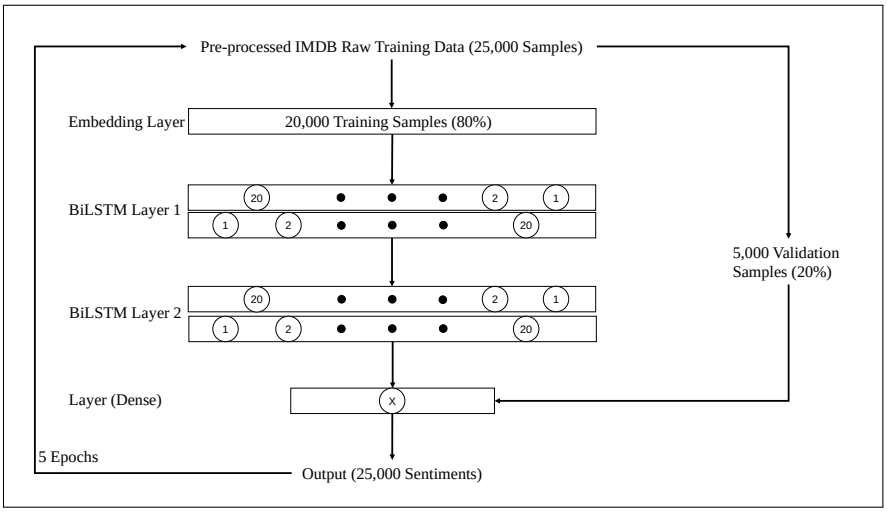

Figure 1: The LSTM model.

As Figure 1 shows, our LSTM model has a bidirectional architecture, similar to Howard and Ruder (2018) model, but it initially contains only 2 BiLSTM layers and 20 units per layer and direction. After the initial embedding layer which

is used for training the word embeddings, both BiLSTM layers are utilized for learning representations. The final, fully-connected dense layer executes the binary classification of the 25,000 training (respectively test) samples with a sigmoid function. As an optimizer the "RMSprop" algorithm (Hinton et al, 2012) is used, as a loss function a binary cross entropy. The number of words used as features is 10,000, and the maximum review length is 500 words. The model is trained for 5 epochs (which is a good number of epochs compared to the results of Hong and Fang (2015) for highly regularized LSTMs) with a batch size of 100, the validation split is 20% (5,000 samples, respectively).

In this model, the following hyperparameters are now to be varied to examine their single impact on the correct classification rate: The number of words considered as features, the sequence length of the comments, the proportion of validation data, the use of pre-trained GloVe word embeddings, the number of hidden BiLSTM layers, the number of units per hidden layer, the dropout and recurrent dropout rates (for preventing overfitting), and the size of the data batches (during training, the training data is divided into batches of a fixed size which are given successively through the network; the weights of the network are updated after every batch). For each hyperparameter, a specific value is set and a variant is selected that suggests a greater learning performance. Within the experiment, only one hyperparameter is chosen into its variant value at the same time. The other hyperparameters stay at their default value. The selected values are summarized in table 3.

The hyperparameters "validation data" and "batch size" were chosen lower in the variant since a larger amount of training data as well as smaller batches suggest a better classification performance. Since an adaptation of the network parameters takes place after each batch, smaller batches mean a higher number of such adjustments and thus deeper learning processes. For all other values, however, a stronger performance is assumed if the values are higher. The values are changed separately while the other hyperparameters maintain their default configuration. The determined values are then compared with the global default variant using the validation data performance in order to show their single impacts on the network performance. In this way, 8 comparison pairs are created (1 for each hyperparameter). If a hyperparameter variation has a positive effect on the validation performance, it will be transformed into a final model which will be compared to the default configuration for investigating whether the effects on accuracy can be cumulated to a high-performing model. The hyperparameter

"dropout" is tested for preventing overfitting during training. At the same time, the machine times are observed. The computations are accomplished with Amazon Web Services (m4.2xlarge, 32GB).

3.4 Results

Model	Default	Variant	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Machine Time
Standard	—	—	94.60%	0.1541	87.84%	0.2905	18 min 20 sec
Max_features	10,000	20,000	95.89%	0.1258	88.68%	0.3106	18 min 7 sec
Max_len	500	1,000	95.49%	0.1326	88.66%	0.2947	18 min 1 sec
Val_split	0.2	0.1	95.18%	0.1395	88.48%	0.2931	19 min 26 sec
GloVe	no	yes	50.23%	0.6932	50.14%	0.6930	16 min
Units / Layer	20	100	94.52%	0.1574	86.12%	0.2992	44 min 17 sec
Layer	2	3	94.98%	0.1421	87.74%	0.3195	26 min 31 sec
Dropout	no	yes	91.65%	0.2275	86.74%	0.3513	21 min 28 sec
Batch_size	100	50	95.10%	0.1431	88.90%	0.2916	28 min 16 sec

Table 3: Training and validation results of the default configuration and the variants.

Without any hyperparameter variation, the default model reaches 94.60% training and 87.74% validation accuracy. The values of the loss function were 0.1541 for training and 0.2905 for validation. Due to overfitting in unregularized BiLSTMs, this value is already reached during the 2nd training epoch. Nevertheless, our model performs on a quite respectful level since there is no pre-training integrated. The training session required 18 minutes and 20 seconds.

The increase of word features (10,000 to 20,000 words) provided 95.89% training and 88.68% validation accuracy (loss function: 0.1258 resp. 0.3106) which means an increase of 0.84% in validation performance compared to the default model. This result was reached in the 2nd epoch as well, another rapid overfitting was observed. The training required 18 minutes and 7 seconds which was surprisingly less than the default model. Since the accuracy rate was higher, this hyperparameter variant was implemented in the final model.

The increase of the maximum sample length (500 to 1,000) also improved the performance (95.49% for training and 88.66% for validation accuracy, 0.1326 resp. 0.2947 for the loss function values), this time an increase of

0.82% in validation accuracy compared to the default model was observed. Not surprisingly, the 2nd training epoch performed best, this variation needed 18 minutes and 1 second training time. This variant was also implemented into the final model.

Changing the ratio of training and validation data from 80:20 to 90:10 resulted in a further increase in validation accuracy to 88.48% (+ 0.64%) which was already achieved during the 2nd epoch (training accuracy: 95.18%; loss function values were 0.1395 for training resp. 0.2931 for validation). Subsequently, overfitting could be observed again. This was accompanied by an increase in computing time to 19 minutes and 26 seconds. Since the accuracy increased due to the greater amount of training data, the final model will also be trained with the higher number of samples.

The use of pre-trained word embeddings from the GloVe database caused a massive loss of accuracy. While computing time was clearly the shortest at precisely 16 minutes, a training accuracy of 50.23% and a validation accuracy of only 50.14% could be achieved (loss function values: 0.6932 for training resp. 0.6930 for validation), which corresponds to a validation accuracy loss of 37.70% compared to the default model. This very poor performance is due to the lack of task-specific training of the word embeddings, which means that the values remained almost constant over the 5 epochs. The strong benefits of pre-training in literature, as found in Howard and Ruder's (2018) model, are achieved through huge datasets used to learn the word embeddings and weights. At the same time, the word embeddings are not frozen, but constantly adapted during the learning process. While the GloVe word embeddings used here is also based on just 400,000 words, for example, WIKITEXT-103 incorporates embedding vectors for about 103,000,000 words. Thus, the word embeddings used are far from having enough information to precisely solve the specific classification task of the IMDB dataset. The pre-trained embedding vectors are therefore not integrated into the final model. However, unsupervised pre-training is indispensable for creating a particularly powerful architecture if it is carried out with very large amounts of useful information and the parameters found are then further adapted to the task.

Increasing the units per hidden layer from 20 to 100 led to a massive increase in computational time to 44 minutes and 17 seconds. This is the consequence of the higher computational effort since the additional units also process a large amount of information during the training. However, the validation accuracy

fell by 1.72% to 86.12% (training: 94.52%) and the values of the loss function were worse as well (0.1574 for training and 0.2992 for validation). This result is particularly surprising given the fact that the most powerful LSTM models from the literature have clearly greater capacities. However, the performance can not be explained by overfitting, since the training data were not classified very well and the best validation performance was not achieved until the 5th epoch. This result indicates additional influences between different hyperparameters, which go beyond separate variations of individual parameters. Due to the inadequate outcome of this study, the final model does not require an increase in the number of units as the higher storage capacity should lead to an increase in classification performance, which was clearly missed.

The integration of a third BiLSTM layer, similar to Howard and Ruder (2018) network, also resulted in a lower validation accuracy of 87.74% (training accuracy: 94.98%) and worse values of the loss function (0.1421 for training and 0.3195 for validation), however, this difference is lower compared to the higher number of units (-0.1% vs default configuration). This ratio was reached during the 3rd epoch so overfitting can be observed another time (presumably by additional network capacity). The machine time increased to 26 minutes and 31 seconds. Although this result does not necessarily preclude the inclusion of a third hidden layer to the final model, due to the increased machine time and the simultaneous (minor) deterioration of the accuracy, the third BiLSTM layer will not be included.

Using a dropout / recurrent dropout regularization with the values 0.2 / 0.2 reduced the validation accuracy of the model by 1.1% to 86.74% (training accuracy: 91.65%) with simultaneous increase of the calculation time to 21 minutes and 28 seconds. The values of the loss function were 0.2275 for training and 0.3513 for validation. However, the dropout was introduced to avoid overfitting and thus increase the stability of the model. Since the top value was reached in the fifth epoch, the dropout was successful so that the regularization is to be evaluated advantageously and integrated into the final model.

The use of a smaller batch size (50 versus 100 samples) brought the highest validation accuracy gain of a single changed hyperparameter (1.06% to 88.90%). The training performance was 95.10% and the values of the loss function were 0.1431 for training and 0.2916 for validation. It is also positive that the validation quota could be reached twice (epoch 2 and 3) before overfitting begins. In this case, the model benefits from a higher number of parameter adjustments

regarding the smaller batch size. However, machine time was quite high at 28 minutes and 16 seconds, due to the smaller denomination of the training data. By increasing the performance, the final model will be trained with smaller batches as well.

Model	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Machine Time
Standard model	94.60%	0.1541	87.84%	0.2905	87.01%	0.3652	18 min 20 sec
Max_features	95.89%	0.1258	88.68%	0.3106	86.54%	0.3603	18 min 7 sec
Max_len	95.49%	0.1326	88.66%	0.2947	86.87%	0,3301	18 min 1 sec
Val_split	95.18%	0.1395	88.48%	0.2931	87.60%	0,3361	19 min 26 sec
GloVe	50.23%	0.6932	50.14%	0.6930	52.81%	1.2817	16 min
Units / Layer	94.52%	0.1574	86.12%	0.2992	86.04%	0.3741	44 min 17 sec
Layer	94.98%	0.1421	87.74%	0.3195	85.90%	0.3415	26 min 31 sec
Dropout	91.65%	0.2275	86.74%	0.3513	85.53%	0.3599	21 min 28 sec
Batch_size	95.10%	0.1431	88.90%	0.2916	87.51%	0.3534	28 min 16 sec
Final model	93.01%	0.1948	88.36%	0.3042	87.46%	0.3779	83 min 28 sec

Table 4: Results of the hyperparameter variants incl. the final model and test data performances.

On the basis of the discussed validation results of the hyperparameter variations, the default configuration should now be modified seeking for a more powerful final model. A total of 5 single hyperparameter variants could be identified as well-working, including the higher number of words considered as features, the larger comment length, the use of smaller batch size, the greater amount of training data, and the integration of dropout regularization to avoid overfitting (as the validation results showed, overfitting in LSTMs is a big issue to deal with). The tested pre-trained GloVe word embeddings, on the other hand, could not be taken into account due to the massive loss of accuracy. Also, the implementation of additional layers and units could not improve the network.

The training of the final model was highly more computationally intensive than the variants of individual hyperparameters (83 minutes and 28 seconds). This observation is not surprising due to the observed calculation times of the individual variations, the computational effort of the individual hyperparameters just adds up in the final model. The accuracy, however, reached 93.10% for training and 88.36% for validation which corresponds to an increase of 0.52% in validation performance compared to the default configuration (reached in the 5th epoch so the dropout implementation was successful in avoiding overfitting). But, at the same time, it is highly noteworthy that the performance in the

validation data is worse than in the variants of the individual positive-acting hyperparameters which were set to improve the network accuracy (dropout regularization was implemented to avoid overfitting). This means that LSTM hyperparameters do not just work on their own but seem to interact with the other hyperparameter settings. In fact, this experimental design is well-suited for understanding the effects of the various hyperparameters on the network in general, but it is not optimal for finding the strongest setting within an LSTM. Nonetheless, the final model has achieved a higher validation performance than the already well-performing default configuration.

For further evaluating the variants and the final model compared to the default configuration, the test dataset of the IMDB dataset was classified. For this, the raw test data was preprocessed as well as the training data (vectorization and word embeddings learned by embedding layer). The default configuration achieved 87.01% test accuracy while the created final model achieved a comparatively stronger accuracy of 87.46%. Compared with the single variants, the separate variation of the validation split and the batch size were even outperforming the final model while the variant of the validation split reached the highest test accuracy (87.60%) with a machine time of 19 minutes and 26 seconds. To summarize, the test classification performance could be increased by 0.45% (resp. approximately 113 additionally correctly classified comments) through varying the 5 hyperparameters classified as positive and by 0.59% (resp. approximately 148 additionally correctly classified comments) through the separate variation of the validation split compared to the default model. The 0.45% increase in classification performance represents an improvement associated with highly increased computational time requirements while the higher increase of 0.59% could be reached with only a small gain of machine time.

4 Discussion / Conclusion

The aim of this work was to investigate the impacts of single hyperparameter variants within an LSTM network to perform the IMDB Large Movie Dataset SA task. For this purpose, an LSTM network based on the task-specific DL models from the literature of recent years was created. A total of 8 hyperparameters contained in this network were separately varied and compared with the default configuration by their validation performance. In this way, 5 hyperparameters (maximum number of words taken into account as characteristics, maximum

comment length, dropout regularization, use of a larger training dataset, and the use of a smaller batch size) could be demonstrated as positive influences while implementing additional hidden layers, additional units per layer and pre-trained GloVe word embeddings could not achieve any positive effects. The variants which improve the validation accuracy were then transformed into a final model to see whether the impacts of the separate hyperparameters could be added. While the validation data performance of the final model was higher than the default model, some single variants outperformed the final model so the effects of single variants were not able to be cumulated. In addition, comparing the default configuration, the separate variants and the final model based on test data accuracy, the default model achieved 87.01% with a machine time of 18 minutes and 20 seconds, while the final model achieved 87.46% at a clearly higher computation time of 83 minutes and 28 seconds. At the same time, the separate variants of the validation split and the batch size even outperformed the final model due to test accuracy and machine time (with the separate variation of the validation split as the overall best configuration performing a test accuracy of 87.60%). In this way, the already precisely classifying default configuration could be increased by a further 0.45% (approximately 113 additional comments) through creating the final model and even 0.59% with a separate variant. In fact, the separate influences of the hyperparameter variants on accuracy could not be cumulated but, at the same time, the machine time did.

Looking at the separate variants, it is striking that the better performance was not achieved by increasing the network capacity (additional layers or units per layer) but the consideration of a larger number of features, longer comments and a larger number of training samples were able to raise accuracy, even the use of smaller batch sizes contributed to a stronger performance. In particular, the network was able to benefit from larger amounts of data and a greater number of iteration steps. At the same time, the results for the variants that result in an increase in capacity (number of units / layers) are surprisingly negative and should not be implemented as a single variant in BiLSTMs which already have a large network capacity. Overall, the results indicate interactions between the various hyperparameters that can not be observed in this experimental setup with separate variants. This is supported by the current literature who use a much higher capacity than the model configured here. Accordingly, higher capacities should definitely not be excluded from the construction of DL models for performing SA, rather such changes should be examined together with other

hyperparameter variants in order to possibly further increase the classification performance.

In spite of the lower classification performance compared to the currently best models, it was possible to clearly demonstrate how the single hyperparameters of an LSTM model influence the performance of the overall architecture. In comparison to the architecture by Howard and Ruder (2018), the performances of the model used here are significantly lower. This is due to the fact that Howard and Ruder (2018) use a huge unlabeled dataset for efficiently pre-training their network. In addition, they combined different hyperparameters for increasing the network capacity (additional BiLSTM layers and more units per layer) while preventing overfitting with dropout regularization. The results of their ULMFiT model indicate interactions between the different hyperparameters as our experiment with separate variants did.

While the separate influences of the hyperparameter variants on overall accuracy could be shown precisely, the experiment has to be limited due to the fact that the validation split during the training epochs has been set randomly so small variances due to different validation samples can not be excluded. Though, since every configuration is trained for 5 epochs with 5 different validation splits, the risk of a variance at the validation results is negligible. Furthermore, no effects between the individual parameter variants were analyzed. These effects could be observed by the surprisingly poor classification results for those variants which increase network capacity and the accuracy of the final model compared to different single hyperparameter variants. In this respect, the investigation is limited, and we would like to encourage further research in the field of hyperparameter variants in LSTM networks. In particular, studies that use this paper as a first step to understand the single hyperparameter effects on the network and go on investigating combinations of variants (i.e. using a fractional factorial design (see, e. g., Gunst and Mason (2009)) can further advance the currently still fragile state of research. We believe that a deeper understanding of hyperparameter influences in LSTMs will definitely help to outperform the current IMDB Large Movie Dataset highscore with new and innovative LSTM models.

References

Aggarwal U, Aggarwal G (2017) Sentiment Analysis: A Survey. International Journal of Computer Sciences and Engineering 5(5):222–225, SivaKumar K (ed).

Ain TQ, Ali M, Riaz A, Noureen A, Kamranz M, Hayat B, Rehman A (2017) Sentiment Analysis Using Deep Learning Techniques: A Review. International Journal of Advanced Computer Science and Applications 8(6):424–433. DOI: 10.14569/IJACSA.2017.080657.

Araque O, Corcuera-Platas I, Sánchez-Rada JF, Iglesias CA (2017) Enhancing deep learning sentiment analysis with ensemble techniques in social applications. Expert Systems with Applications 77:236–246. DOI: 10.1016/j.eswa.2017.02.002.

Arel I, Rose DC, Karnowski TP (2010) Deep Machine Learning - A New Frontier in Artificial Intelligence Research. IEEE Computational Intelligence Magazine 5(4):13–18. DOI: 10.1109/MCI.2010.938364.

Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2):157–166. DOI: 10.1109/72.279181.

Brychcín T, Habernal I (2013) Unsupervised Improving of Sentiment Analysis Using Global Target Context. Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP) 2013, pp. 122–128, Mitkov R, Angelova G, Bontcheva K (eds), INCOMA Ltd. Shoumen.

Cambria E, Schuller B, Xia Y, Havasi C (2013) New Avenues in Opinion Mining and Sentiment Analysis. IEEE Intelligent Systems 28(2):15–21. DOI: 10.1109/MIS.2013.30.

Dai AM, Le QV (2015) Semi-supervised Sequence Learning. In: Advances in Neural Information Processing Systems 28, Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds). Curran Associates, Inc., pp. 3079–3087. URL: http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.

Deng L (2011) An Overview of Deep-Structured Learning for Information Processing. Proceedings of the Asian-Pacific Signal & Information Processing Annual Summit & Conference (APSIPA-ASC) 2011, pp. 1–14, EI Compendex, Xi'an, China. URL: http://www.apsipa.org/proceedings_2011/pdf/APSIPA336.pdf.

Deng L (2014) A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing 3:1–29, Cambridge University Press, Cambridge. DOI: 10.1017/ATSIP.2013.9

Dieng AB, Wang C, Gao J, Paisley J (2016) TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. arXiv 2016(11):1–13. URL: http://arxiv.org/pdf/1611.01702v2.

Gal Y, Ghahramani Z (2015) A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. arXiv 2015(12):1–14. URL: http://arxiv.org/

pdf/1512.05287v5.

Gers FA, Schmidhuber J, Cummins F (1999) Learning to forget: continual prediction with LSTM. 9th International Conference on Artificial Neural Networks: ICANN '99, pp. 850–855, Edinburgh, United Kingdom. DOI: 10.1049/cp:19991218.

Google (2013) word2vec. URL: https://code.google.com/archive/p/word2vec/ [accessed 02.11.2018].

Gray S, Radford A, Kingma DP (2017) Gpu kernels for block-sparse weights. arXiv 2017(11):1–12. URL: http://arxiv.org/pdf/1711.09224..

Gunst RF, Mason RL (2009) Fractional factorial design. Computational Statistics 1(2):234–244. DOI: 10.1002/wics.27..

Hinton G, Srivastava N, Swersky K (2012) Neural Networks for Machine Learning. video lectures Coursera, pp. 1–31.

Hochreiter S (1991) Untersuchungen zu dynamischen Neuronalen Netzen. Diploma thesis, Institut für Informatik, München, Technische Universität München.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hong J, Fang M (2015) Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts. Stanford University Technology Report, pp. 1–9. URL: https://cs224d.stanford.edu/reports/HongJames.pdf.

Howard J, Ruder S (2018) Universal Language Model Fine-tuning for Text Classification. arXiv 2018(05):1–12. URL: http://arxiv.org/pdf/1801.06146v5.

Jain AK, Mao J, Mohiuddin KM (1996) Artificial neural networks: A tutorial. Computer 29(3):31–44. DOI: 10.1109/2.485891.

Johnson R, Zhang T (2015) Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. Advances in Neural Information Processing Systems 28 (NIPS 2015), pp. 919–927, Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds), Montreal, Canada.

Johnson R, Zhang T (2016) Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings. arXiv 2016(05):1–9. URL: http://arxiv.org/pdf/1602.02373v2.

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. Proceedings of the 31st International Conference on Machine Learning (ICML) 32:1188–1196, Xing EP, Jebara T (eds), JMLR.org, Beiing, China.

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. DOI: 10.1038/nature14539.

Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning Word Vectors for Sentiment Analysis. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies 1:142–150, Association for Computational Linguistics, Portland, Oregon, USA.

Maynard D, Funk A (2011) Automatic Detection of Political Opinions in Tweets. Extended Semantic Web Conference (ESWC) 2011: ESWC 2011 Workshops, pp. 88–99,

Garcia-Castro R, Fensel D, Antoniou G (eds), Heraklion, Greece. DOI: 10.1007/978-3-642-25953-1.

Medhat W, Hassan A, Korashy H (2014) Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal 5(4):1093–1113. ISSN: 2090-4479, DOI: 10.1016/j.asej.2014.04.011.

Merity S, Keskar NS, Socher R (2017) Regularizing and Optimizing LSTM Language Models. arXiv 2017(08):1–10. URL: http://arxiv.org/pdf/1708.02182v1.

Mesnil G, Mikolov T, Ranzato M, Bengio Y (2014) Ensemble of Generative and Discriminative Techniques for Sentiment Analysis of Movie Reviews. arXiv 2014(05):1–5. URL: http://arxiv.org/pdf/1412.5335v7.

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient Estimation of Word Representations in Vector Space. arXiv 2013(09):1–12. URL: http://arxiv.org/pdf/1301.3781v3.

Miyato T, Dai AM, Goodfellow I (2016) Adversarial Training Methods for Semi-Supervised Text Classification. arXiv 2016(05):1–11. URL: http://arxiv.org/pdf/1605.07725v3.

Narayanan V, Arora I, Bhatia A (2013) Fast and accurate sentiment classification using an enhanced Naive Bayes model. International Conference on Intelligent Data Engineering and Automated Learning (IDEAL) 2013, pp. 194–201, Yin H, Tang K, Gao Y, Klawonn F, Lee M, Weise T, Li B, Yao X (eds), Hefei, China.

Pennington J, Socher R, Manning C (2014) Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, Moschitti A, Pang B, Daelemans W (eds), Association for Computer Linguistics, Doha, Qatar. DOI: 10.3115/v1/d14-1162.

Radford A, Jozefowicz R, Sutskever I (2017) Learning to Generate Reviews and Discovering Sentiment. arXiv 2017(04):1–9. URL: http://arxiv.org/pdf/1704.01444v2.

Ravi K, Ravi V (2015) A survey on opinion mining and sentiment analysis: Tasks, approaches and applications. Knowledge-Based Systems 89:14–46. ISSN: 0950-7051, DOI: 10.1016/j.knosys.2015.06.015.

Rui W, Liu J, Jan Y (2016) Unsupervised feature selection for text classification via word embedding. Proceedings of the 2016 IEEE International Conference on Big Data Analysis (ICBDA), pp. 1–5, Hangzhou, China.

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536. DOI: 10.1038/323533a0.

Schmidhuber J (2015) Deep Learning in Neural Networks: An Overview. Neural Networks 61:85–117. ISSN: 0893-6080, DOI: 10.1016/j.neunet.2014.09.003.

Schuster M, Paliwal KK (1997) Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing 45(11):2673–2681. DOI: 10.1109/78.650093.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1):1929–1958. DOI: 10.5555/2627435.2670313.

Stanford (2014) GloVe. URL: https://nlp.stanford.edu/projects/glove [accessed 02.11.2018].

Turney PD (2002) Thumbs up or thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th annual meeting on association for computational linguistics (ACL), pp. 417–424, Isabelle P, Charniak E, Lin D (eds), Association for Computer Linguistics, Philadelphia, Pennsylvania, USA. DOI: 10.3115/1073083.1073153.

Wang S, Manning CD (2012) Baselines and bigrams: Simple, good sentiment and topic classification. Proceedings of the 50th annual meeting of the Association for Computational Linguistics 2:90–94, Li H, Lin CY, Osborne M, Lee GG, Park JC (eds), Association for Computer Linguistics, Jeju Island, Korea. URL: https://www.aclweb.org/anthology/P12-2018.

Xu W, Sun H, Deng C, Tan Y (2017) Variational Autoencoder for Semi-Supervised Text Classification. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), pp. 3358–3364, San Francisco, California, USA. URL: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14299.

Yadollahi A, Shahraki AG, Zaiane OR (2017) Current State of Text Sentiment Analysis from Opinion to Emotion Mining. ACM Computing Surveys 50(2):1–33. DOI: 10.1145/3057270.

Yang Y, Pedersen JO (1997) A Comparative Study on Feature Selection in Text Categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, ICML '97, pp. 412–420. ISBN: 15-5860-486-3, DOI: 10.5555/645526.657137.

Zhang L, Wang S, Liu B (2018a) Deep learning for sentiment analysis: A survey. Data Mining and Knowledge Discovery 8(4):e1253. DOI: 10.1002/widm.1253.

Zhang Q, Yang LT, Chen Z, Li P (2018b) A survey on deep learning for big data. Information Fusion 42:146–157. ISSN: 1566-2535, DOI: 10.1016/j.inffus.2017.10.006.

Zhang X, Zhao J, LeCun Y (2015) Text Understanding from Scratch. arXiv 2015(02):1–9. URL: http://arxiv.org/pdf/1502.01710v5.

Zhou ZH, Feng J (2017) Deep Forest. arXiv 2017(02):1–34. URL: http://arxiv.org/pdf/1702.08835v3.