10-701 Project Final Report
IMDB Text Classification

Boyuan Lai	Daanish Ali Khan	Ibrahim El Shar	Utku Tarik Bilgic
bol27@pitt.edu	malikhan@andrew.cmu.edu	ije8@pitt.edu	utb3@pitt.edu

1 Introduction

In this study, we propose a novel text representation for language classification problems. The models and techniques presented were evaluated using the IMDB Movie Review Dataset. Our aim is to design and implement a binary classifier to predict a movie review sentiment as positive or negative. The dataset is composed of 50k of movie reviews from the IMDB website, half of them are positive (having a rating ≥ 7), other half is labeled as negative (having a rating ≤ 4).

The input to our classifier will be raw english text, and our output will be the class probability/prediction. The raw text will have to be converted into a representation that can be processed computationally. Existing techniques, traditional and neural-net based, rely on word-level representations of the data. Our hypothesis is that word-level representations may not sufficiently capture nuanced contextual information from the text. We believe that sentence-level representations may capture better inter-word dependencies, and yield potentially better results.

In this report, we present multiple novel sentence-based text representations, building on state-of-the-art Natural Language Processing (NLP) techniques. Furthermore, we evaluate their efficacy as sentiment representations using a variety of traditional binary classifiers and analyze the relative performance on the sentiment classification task.

2 Background

We chose to implement and evaluate two different baseline architectures: a traditional baseline and a neural-net baseline. The traditional baseline was a Naive Bayes classifier (see for e.g. [1]) trained on a Bag-of-Words representation of the data. The vocabulary was filtered of stop-words, and sorted according to decreasing order of frequency. The Naive Bayes classifier was then iteratively trained on an increasing number of feature-words, starting with the 100 most frequent, increasing the number of feature-words by 100 each time, up to a total of 10,000 feature words. This baseline was implemented using the Natural Language Tool-Kit (NLTK)[2]

The classification accuracy on the test set of the Naive Bayes classifier is shown in Figure 1. The test accuracy is represented in the y-axis, while the number of feature-words (x most frequent words) is represented by the x-axis. The highest test-accuracy achieved by Naive Bayes was 84.75%.

The neural-net baseline was a bidirectional multi-layer Long Short Term Memory (LSTM)[3] network. Each word in the input sentence was passed through an embedding layer to produce a 100-dimensional word-vector. The sequence of word-vectors was then passed through a 3-layer Bi-LSTM, where the hidden unit at the last timestep was passed through a projection layer, with one output neuron with sigmoid activation, to produce the class probability. In order to leverage pre-trained word-level representations, our embedding layer was initialized as a pretrained GLOVE embedding [4]. For reguralization, dropout of 0.25 was used. The network was trained using the ADAM optimizer[5] and learning-rate decay, and was implemented using the PyTorch neural-net framework.

The training statistics of the neural-net baseline is depicted in Figure 2 (with a $70 - 30$ split of the original training set). Figure 2a shows the training and validation Binary Cross Entropy Loss on the

Preprint. Work in progress.

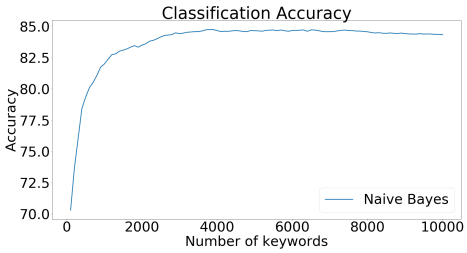
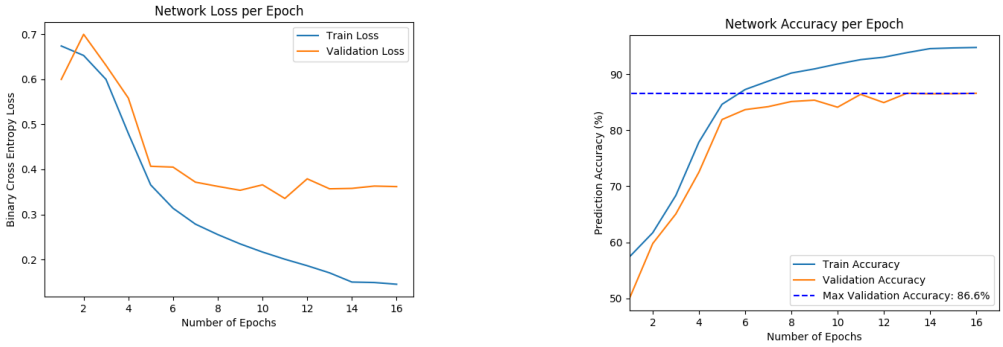


Figure 1: Naive Bayes baseline



(a) Training and Validation Binary Cross Entropy Loss

(b) Training and validation accuracy

Figure 2: Baseline - Bi-LSTM Performance

y-axis, and epoch number on the x-axis. Figure 2b shows the training and validation accuracy. The highest test-accuracy achieved by this model was 87.1%.

From Fig. 1, it can be seen that Naive Bayes classifier achieves about 84.7% accuracy with a bag of words of size ≥ 3700. At this point, increasing the size of feature words does not help anymore in increasing the accuracy of the classifier. Nonetheless, we think that an accuracy of 84.7% is good enough and that our Naive Bayes baseline has a reasonable accuracy in classifying the text of IMDB Movie Review dataset. The neural-net baseline performed as expected, providing a marginal accuracy improvement over the Naive Bayes classifier. The network was trained using a train and validation set built from the original training set ($70 - 30$ split). From Figure 2a, we can see that the training loss decreases smoothly until it saturates after 16 epochs of training.

3 Related Work

The best performing systems on this dataset are neural-network based techniques. Language-Model based systems like ULMFit[6] use Recurrent Neural Networks (RNNs) that are trained initially as language models on large corpora. The pretrained language model is then fine-tuned on the IMDB dataset[7] for binary classification, achieving an error rate of 4.6. At the time of this report, this is the lowest published error rate on this dataset.

For general-purpose sentence classification, Multi-Task Deep Neural Networks (MTDNNs)[8], learn sentence representations by training the model on a series of NLP and NLU tasks (sentence similarity, relevance ranking, etc.) before being fine-tuned on sentence classification. While this technique has not been officially evaluated on the IMDB dataset[7], it achieves high accuracies on other sentence classification tasks. Learning sentence representations as embeddings is an active area of research in deep learning for NLP[9, 10]. Convolutional Neural Network (CNN) based techniques have been used to dynamically model sentences to learn representations[11].

Techniques using Attention [12] have been applied to sequence modelling tasks successfully, but are better suited for sequence-to-sequence language tasks like speech recognition and machine translation.

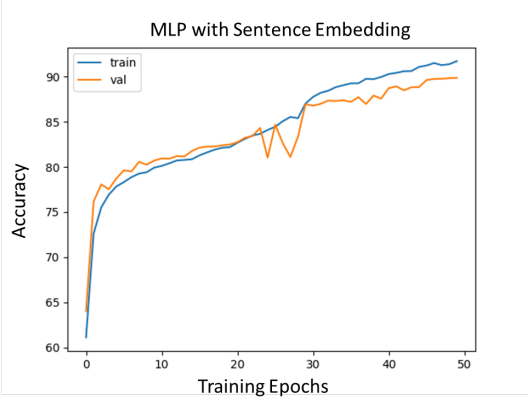


Figure 3: Training and Validation Accuracy for MLP with Sentence Embedding as Whole Review

More recent models dubbed *Self Attentive Models*[13, 14] have performed extremely well on general NLP tasks. These models allow the network to identify significant portions of the input, improving the model's abilities to learn complex relationships in sentences.

Prior to the advent of RNNs and Attention, traditional machine learning techniques, like Naive-Bayes-SVMs [15], were the state of the art, with an error rate of 8.78% on the IMDB dataset [7]. Another traditional technique for sentiment classification is DeepForest [16], which uses a deep random forest in a cascading structure to achieve an error rate of 10.8% on IMDB[7], which at that time was state of the art. Both these techniques significantly outperformed prior deep learning based classifiers.

4 Methods

In order to learn a better representation of the input text, we evaluated multiple models. All the representations are based on the neural-network sentence-embedding concept as described in prior work [9].

4.1 NLI Paragraph Embedding

The first approach utilized a neural network that was pre-trained to perform Natural Language Inference (NLI) tasks on the SNLI Dataset. The architecture of this network can be seen in Figure 4

The input text is first converted into word embeddings using GloVe, then the resulting sequence of 300-dimensional vectors is passed through a bi-directional, two layer GRU (pre-trained for NLI) with a hidden size of 2048. The final hidden states from each direction are concatenated and passed through a Multi-Layer Perceptron (MLP) where the output layer has one neuron with the sigmoid activation.

The loss function for this network was the Binary Cross Entropy Loss, defined as:

$$BCE(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

The network was optimized using Adam[5], with learning-rate decay every 10 training iterations. Adaptive Moment Estimation(Adam) optimizer is a stochastic gradient descent (SGD) based optimization algorithm, which uses estimates of first and second order moments for computing individual adaptive learning rates for different parameters. SGD is an optimization algorithm that uses a random sample instead of the entire data set to compute an estimate of the gradient in each iteration. In Adam, in an iteration, first the biased first and second moments are updated and then their bias-corrected estimates are computed. Backpropagation uses the chain rule to calculate the gradient of the objective function in a neural network with respect to its weight and bias parameters.

We first try a "naive" approach where one embedding is generated for each review. These embeddings are then classified using a Multilayer perceptron (MLP). Training and validation accuracy is given in Figure 3. Surprisingly, using these embeddings our test accuracy barely improved; yielding a

Figure 4: Paragraph Embedding Network

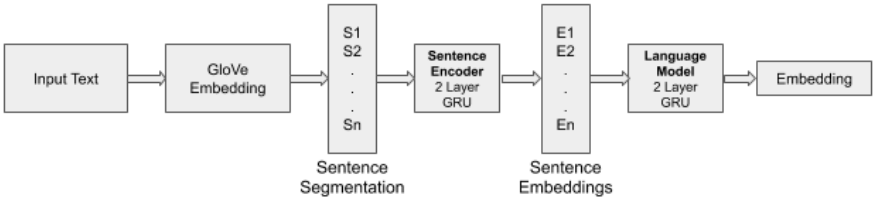


Figure 5: Segmented Sentence Embedding

validation accuracy of 90%, but a test accuracy of 88.5%, a marginal improvement from the Bi-LSTM baseline. We attribute this shortfall of this representation to its inadequacy of capturing long-term dependencies. When we analyze the results, we have seen that for the longer reviews(with more than 300 words), test accuracy is 79.2%. However, for the shorter reviews(with 300 words or less), accuracy is 91.6%.

To address this problem, we attempt to decompose each review into its sentences and work with those, which will help us to capture the dependencies and relationships among the sentences in a review.

4.2 Segmented Sentence Representation

Another approach we evaluated, segments the input text into sentences before embedding each using a sentence encoder. Our hypothesis was that it would be much easier to learn long-term dependencies using the segmented approach as dependencies that were previously separated by hundreds of words, will now be captured between a few sentences.

The input to the network is the review text, that is first encoded using GloVe, and then segmented into sentences using regular expression matching. Each individual sentence is encoded using the same pretrained GRU architecture from Section 4.1, resulting in a sequence of sentence embeddings. This sequence of embeddings was used to pre-train a (separate) GRU network as a Language Model (LM). In this stage, the LM network is given a sequence of sentence embeddings as input, and is trained to predict the next sentence embedding.

After the GRU has processed every sentence-embedding, the final hidden representation of the GRU is used as the representation of the input text. This yields a single fixed length vector that encodes the entire text, regardless of the size of the input. We can use the embeddings produced by our network to train various traditional binary classification models, but the embedding has a dimensionality of 4096. In order to reduce the dimension, we decided to performed Principal Component Analysis on the data. (The embeddings were normalized so that they had 0 mean and unit variance.)

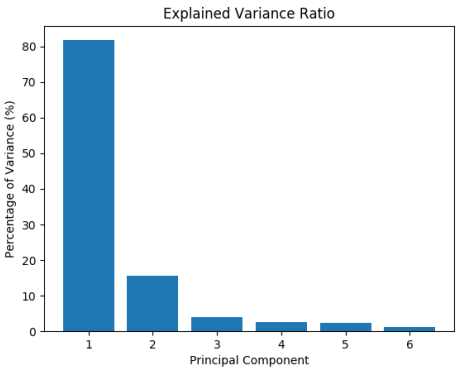


Figure 6: Variance Ratios of Principal Components

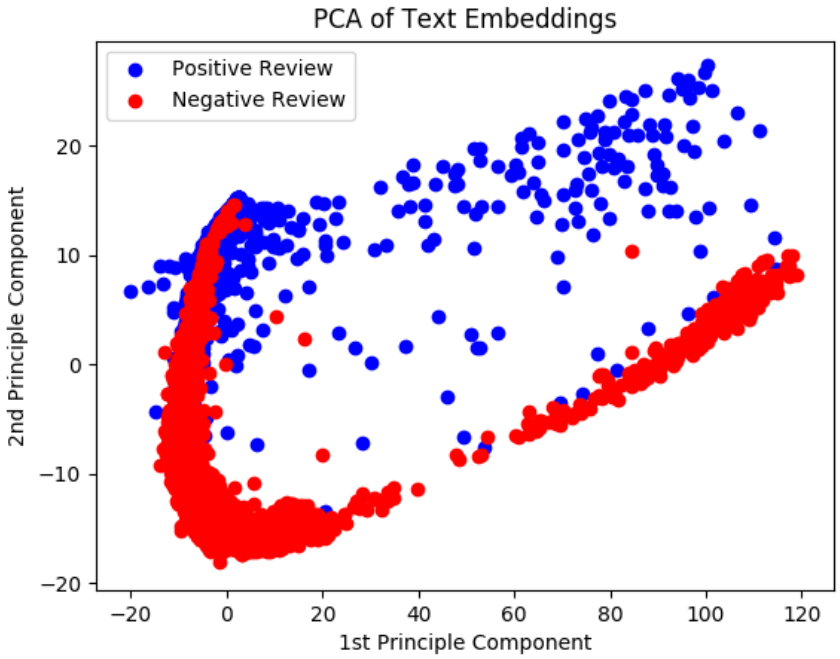


Figure 7: PCA of the Encoded Data

The first two principal components account for more than 95% of the variance in the data (Figure 6), thus we projected the entire dataset onto the first two principal components of the train-set. The first two principal components are visualized in Figure 7. The results are reported in Table 1

4.3 End-to-End Segmented Network

In order to maximize the performance of our text embeddings, we combined multiple levels of representation learning into one end-to-end model. The input text is encoded using a pre-trained GloVe network, following which it is segmented into the sentence representation. Each *sentence* is then encoded using a pre-trained sentence encoder, producing a sequence of sentence embeddings, which is then passed through a pretrained encoder, from Section 4.2, to produce a fixed length vector representation of the whole text.

The network is fine-tuned on the binary sentiment classification task, where the final embedding is then passed through an MLP with a hidden size of 512 ans one output neuron with a sigmoid activation. The network architecture can be seen in Figure 8 The entire network is trained on the IMDB dataset, including the word embedding layer, the sentence encoder, the sentence language model, and the final MLP.

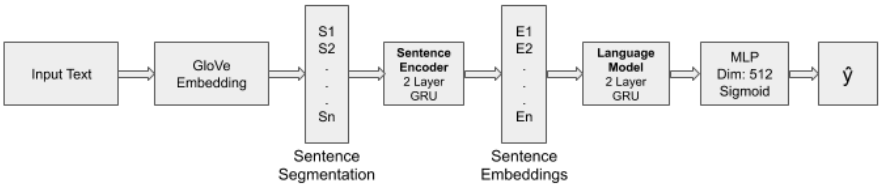


Figure 8: End-to-End Architecture

To train this network, we used the Binary Cross-Entropy Loss described in Section 4.1, with the Adam optimizer.

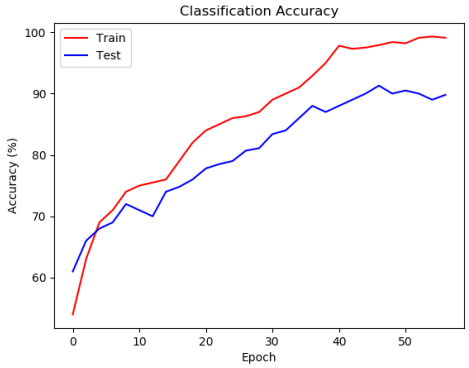


Figure 9: End-to-End Network Accuracy

The accuracy plot is shown in Figure 9. This model achieves a test accuracy of 91.3%, a significant improvement from the other proposed representations.

5 Results

The overall performance of the discussed models is presented in Figure 10. The best performing classifier is the end-to-end segmented network, with an overall accuracy of 91.3%.

Table 1 compares the performance of the classifiers using the precision, recall, and F1 score. The precision, or positive predictive value, measures the ratio of the number of true positives to the number of predicted positives. The recall, or sensitivity, is the ratio of the number of true positives to the actual number of positives in the dataset. The F1 metric is the harmonic mean of the precision and recall. It is commonly used as the metric for classification datasets that have balanced class distributions.The F-1 score can be used for the IMDB dataset due to the fact that there are an equal number of positive reviews and negative reviews (25k).

	Test Acc.	Precision	Recall	F1-score
Logistic Regression	0.8496	0.87	0.85	0.85
Decision Tree	0.8714	0.88	0.87	0.87
Random Forest	0.8708	0.88	0.87	0.87
SVD(rbf kernel)	0.8834	0.89	0.88	0.88
SVD(linear kernel)	0.8537	0.88	0.85	0.85
Gaussian NB	0.8518	0.88	0.85	0.85
K-NN(k=1)	0.8779	0.88	0.88	0.88
K-NN(k=5)	0.8830	0.88	0.88	0.88
K-NN(k=10)	0.8796	0.88	0.88	0.88
AdaBoost	0.8606	0.88	0.86	0.86
Radius-NN(r=10)	0.8460	0.87	0.85	0.85
End-to-End Encoder	**0.913**	**0.90**	**0.90**	**0.90**

Table 1: Performance of classifiers with sentence embeddings

The results of experiments largely align with our expectations. We expected the final model to offer the highest performance across our performance metrics due to the additional training each feature extractor received. Initially we expected a significant improvement over the baseline LSTM when using the sentence embedding network, but the actual performance was only a marginal improvement.

Our results did not reach the state of the art performance, but the results from the end-to-end network are comparable with state of the art models for the dataset from 3-5 years ago.

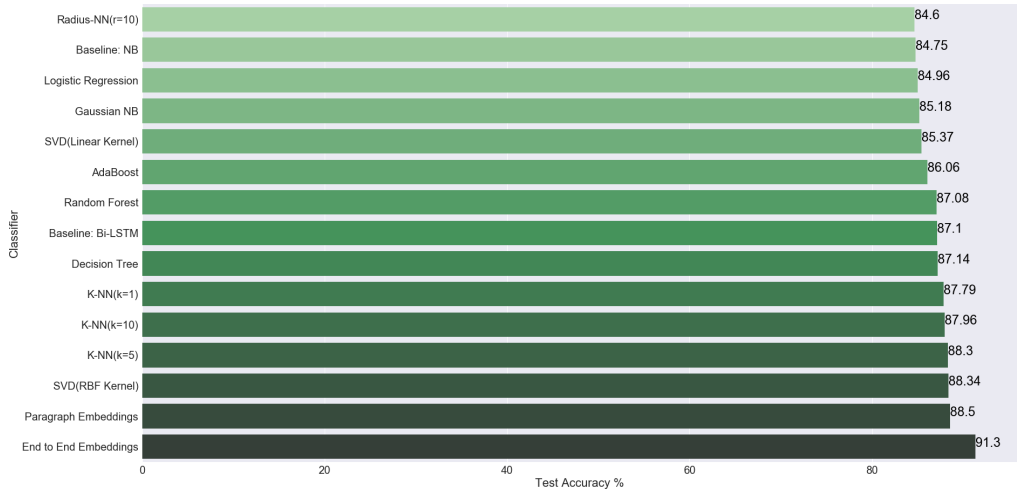


Figure 10: Bar plot comparing the test accuracies of all the methods implemented herein including the baselines. Note that all the methods other than the baselines use sentence embeddings.

6 Discussion

The end-to-end model was able to significantly improve upon the performance of the (un-segmented) sentence-embedding network. We believe this is due to the novel combination of multiple stages of text representation. Experiments with the end-to-end model *without* pre-training any of the intermediate representations resulted in sub-optimal performance. This strongly indicates that initial pre-training of each intermediate representation separately is key to the model's performance.

Additionally, this model's performance appears to validate the natural assumption that an information-rich representation of text can be built from a collection of good sentence embeddings; and a good sentence embedding can be built from a sequence of good word embeddings.

This model can be significantly improved. Due to computational limitations and lack of access to high performance GPUs, we were not able to train the representation on any large corpora. We

believe that a similar architecture, that is sufficiently parameterized and trained on much more data, can continue to improve the quality of the embeddings.

Furthermore, in this study we evaluated our models on Sentiment Classification, but this architecture can be scaled to any text-based task in machine learning. Increasing the number of tasks that the intermediate representations are pre-trained on will improve the generalisability of the embeddings, as will training on larger corpora.

References

[1] Tom M Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression. *Machine learning*, pages 1–17, 2010.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[3] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[4] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[7] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[8] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[9] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[10] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[15] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[16] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835*, 2017.