Text Classification using Different Feature Extraction Approaches

Robert Dzisevič
Department of Information Technology
Vilnius Gediminas Technical University
Vilnius, Lithuania
robert.dzisevic@gmail.com

Dmitrij Šešok
Department of Information Technology
Vilnius Gediminas Technical University
Vilnius, Lithuania
dmitrij.sesok@vgtu.lt

***Abstract*—In this paper, we examine the results of applying three different text feature extraction approaches while classifying short sentences and phrases into categories with a neural network in order to find out which method is best at capturing text features and allows the classifier to achieve highest accuracy. The examined feature extraction methods include a plain Term Frequency Inverse Document Frequency (TF-IDF) approach and its two modifications by applying different dimensionality reduction techniques: Latent Semantic Analysis (LSA) and Linear Discriminant Analysis (LDA). The results show that the TF-IDF feature extraction approach outperforms other methods allowing the classifier to achieve highest accuracy when working with larger datasets. Furthermore, the results show that the TF-IDF in combination with LSA approach allows the classifier to achieve similar accuracy while working with smaller datasets.**

Keywords—text classification, neural network, feature extraction, term frequency inverse document frequency, latent semantic analysis, linear discriminant analysis

I. Introduction

Text classification is a foundational component in many natural language processing applications, such as email, information filtering, sentiment analysis, search engines [2] and even customer interactions using virtual agents – chatbots [14].

The main goal of text classification is to assign the text document to one or more predefined categories based on its content. The task can be solved using various machine learning models but before applying any of them – information needs to be extracted from the text message. This process is called text feature extraction and it is one of the core steps in text classification, which allows to retrieve representations of text messages [7], [12]. During this step, uncorrelated and superfluous text features are removed. In addition, this step can also reduce the dimension of feature space, thus enabling to reduce the amount of data to be analyzed by removing unnecessary features. As part of data preprocessing, this step can also improve the accuracy of a learning model [12].

This paper outlines the most common approaches used in text feature extraction and examines the behavior and results of a neural network classifier by applying different feature extraction methods.

Neural network is a machine learning model which is widely used in natural language processing problems: language modelling [18], machine translation [20], pos-tagging [9], caption generation [21], text summarization [17], question answering [5].

II. Feature Extraction

Text feature extraction holds a crucial role in text classification because it directly influences the classification accuracy. Feature extraction is based on vector space model, where a text is viewed as a dot in a N-dimensional space. Each dimension of the dot represents one feature of the text in digital form. Feature extraction algorithms usually use a keyword set. Based on these predefined keywords, the feature extraction algorithm calculates weights of the words in the text and then forms a digital vector which is the feature vector of the text [11].

The application of Term Frequency Inverse Document Frequency (TF-IDF) weighing scheme is one of the approaches to extract features [12]. Although in most cases it is simple and effective, this approach has its own drawbacks. If the text corpus is large, this method can generate feature vectors with a large number of dimensions, which potentially could increase the chances to overfit the classification model. This can be solved by applying various dimensionality reduction techniques. The idea behind this approach is that the reduced dimensions should be representing something more close to the concepts about which the document is referring to. One of the commonly used dimensionality reduction techniques in natural language processing are Latent Semantic Analysis (LSA) and Linear Discriminant Analysis (LDA). In the following paragraphs, three different feature extraction approaches are described: TF-IDF, TF-IDF in combination with LSA and TF-IDF enhanced with LDA.

The TF-IDF weighing scheme works by weighing a keyword in any context and assigning the importance to that keyword based on the number of times it appears in the document. Furthermore, this information retrieval technique checks how relevant the keyword is throughout the corpus [16]. The overall approach of this feature extraction method works as follows. Given a document collection D, a word w and an individual document $d \in D$, we calculate:

$$w_{i,j} = tf_{i,j} \times \log(N / df_i), \tag{1}$$

here $w_{i,j}$ is the weight for term i in document j, N is the number of documents in the corpus, df_{ij} is the term frequency of term i in document j and df_i is the document frequency of term i in the corpus. The idea of TF-IDF is that words in a document can be divided into two classes: those words which are unique and those words which are not unique, whether the term is relevant to the topic of a document or not [15].

One of the main drawbacks is the size of the feature set in TF-IDF for text data which equals to the size of the vocabulary

978-1-7281-2499-5/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: University of Limerick. Downloaded on August 03,2025 at 14:02:57 UTC from IEEE Xplore. Restrictions apply.

across the entire corpus, thus resulting in huge computation on weighing all the words in the data set [16].

LSA is a technique which is used to analyze relationships between a set of documents and the words they contain by generating a set of concepts related to the documents. The LSA algorithm assumes that terms that are close in meaning are tend to occur in similar places of text. A matrix containing word counts or weights per document (TF-IDF weights matrix) is constructed from a large text corpus and later the Singular Value Decomposition (SVD) technique is used to reduce the number of dimensions while preserving the similarity structure in the data [10].

LDA is a method used in machine learning to find a linear combination of features that characterizes or separates two or more classes of objects. The resulting combination can be used as a linear classifier or for dimensionality reduction [6].

The generalized LDA approach can be summarized in the following steps. Firstly, the *d*-dimensional mean vectors for different classes are calculated. Secondly, the scatter matrices (in-between-class and within-class matrix) are calculated. Later the eigenvectors and corresponding eigenvalues for the scatter matrices are calculated. At the next step the eigenvectors are sorted by decreasing eigenvalues and k (where $k \le n-1$) eigenvectors with the largest eigenvalues in order to form a $k \times d$ dimensional matrix W. Finally, the W matrix is used to transform the samples onto the new subspace. The transformation can be summarized by this equitation:

$$Y = X \times W, \tag{2}$$

here X is a $n \times d$ is the dimensional matrix which represents the n samples and Y are the transformed $n \times k$ is the dimensional samples.

III. Data Preparation and Implementation

For our numerical experiments we constructed a dataset of 10000 advertisement headers from Lithuanian advertisement websites *alio.lt* and *skelbiu.lt.* The advertisements were classified into 20 categories. The data was gathered by web scraping the advertisement website using a custom written script in *Python* programming language.

In addition to data preparation, some data noise reduction techniques were applied: punctuation marks, special and numerical characters were removed, most commonly used words in the language (stop words) were also removed from the dataset (see Fig. 1). The stop word list was taken from a Github repository [3]. Finally, the data set was normalized by applying stemming on each word in the data set. The stemming was performed by utilizing the stemming algorithm for Lithuanian language found in the Snowball compiler project [13].

The whole implementation was done in Python programming language using Jupyter Notebook development environment. The text feature extractors were built by utilizing scikit-learn library.

The neural network classifier was implemented by using *keras* and *tensorflow* libraries. The classifier was implemented as a feedforward neural network consisting of two fully connected neuron layers.

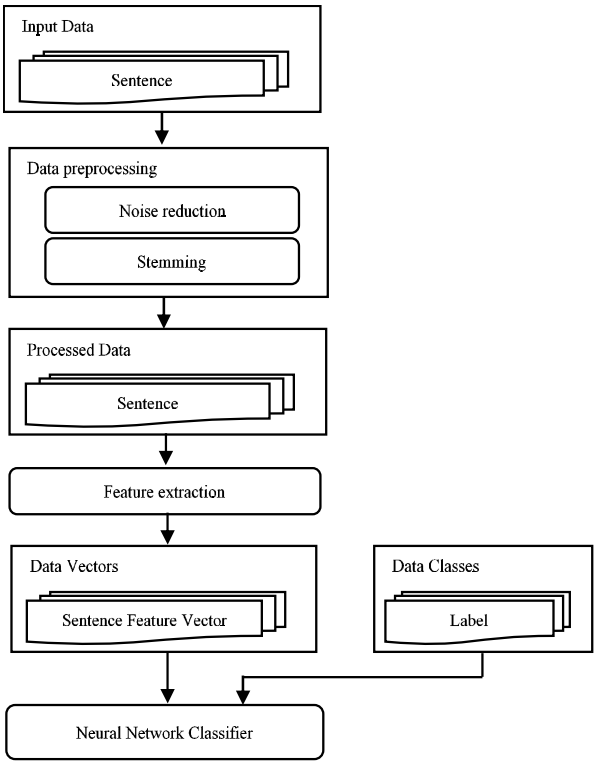


Fig. 1. Text data processing flow.

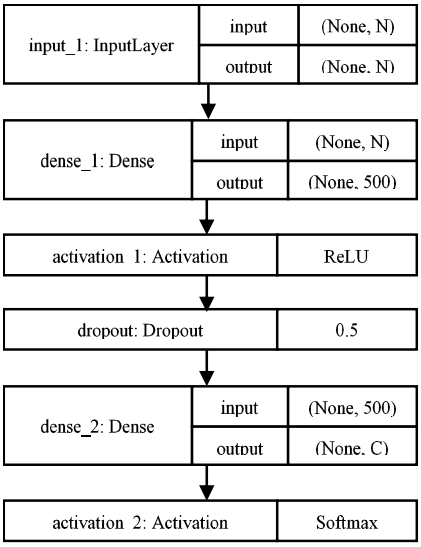


Fig. 2. Architecture of a neural network model.

In Fig. 2 *input_1* is the layer containing training data where each row is of dimension *N*. *Dense* layers are fully connected layers of neurons in which every input is connected to every output by a weight. Each fully connected layer has an activation function which essentially normalizes the output before it is passed on to the next layer of neurons. Layer *dense_1* uses Rectified Linear Units (ReLU) as an activation function [1]:

$$R(z) = \max(0, z). \tag{3}$$

Layer *dense_2* uses the Softmax function for activation [1]:

Authorized licensed use limited to: University of Limerick. Downloaded on August 03,2025 at 14:02:57 UTC from IEEE Xplore. Restrictions apply.

$$\sigma(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \quad (4)$$

The *Dropout* layer is a method used to prevent overfitting the classifier. This technique randomly sets a fraction of input units to zero at each update during training. In Fig. 2 the specified fraction is equal to 0.5 [19].

The last layer (*dense_2*) outputs vectors of dimension *C* which is equal to the number of categories we want to classify. Each vector in the last layer output shows the probabilities of an input vector belonging to each possible category.

As a loss function the classifier's model uses the Categorical Cross-Entropy function:

$$H(y, p) = -\sum_i y_i \log(p_i). \quad (5)$$

The loss function basically evaluates how well the model performs the classification [4].

As the model's optimizer the AdaMax algorithm is used. The optimizer is used to bind together the loss function and the model parameters by adjusting the weights of neurons in the model in response to the output of the loss function [8].

The number of parameters in the first fully connected layer (*dense_1*) is equal to:

$$Q(N) = N \times 500+500, \quad (6)$$

here *N* is equal to the number of dimensions in an input vector.

The number of parameters in the last fully connected layer (*dense_2*) is equal to:

$$Q(C) = C \times 500+C, \quad (7)$$

here *C* is equal to the number of possible categories to classify.

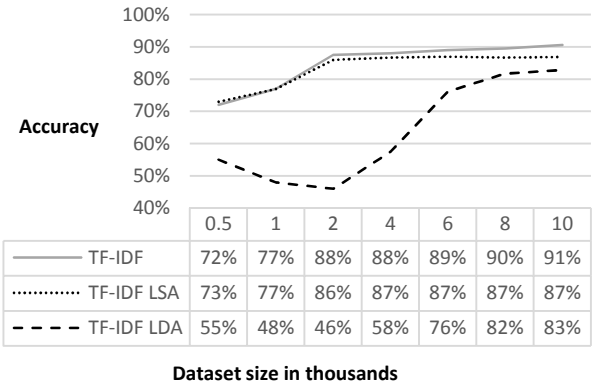
IV. Evaluation

Three different text feature extraction techniques were applied: TF-IDF, TF-IDF LSA and TF-IDF LDA. The classifier's performance was evaluated by feeding the classifier datasets of different sizes (500, 1000, 2000, 4000, 8000, 10000). In each experiment the classifier's training was performed on 80% of the data and validation on 20% of the data. The total training duration for each experiment was 20 epochs. The experiments were performed on an *Apple MacBook Pro 13* with an *Intel Core i5 2.9 GHz* processor.

V. Results

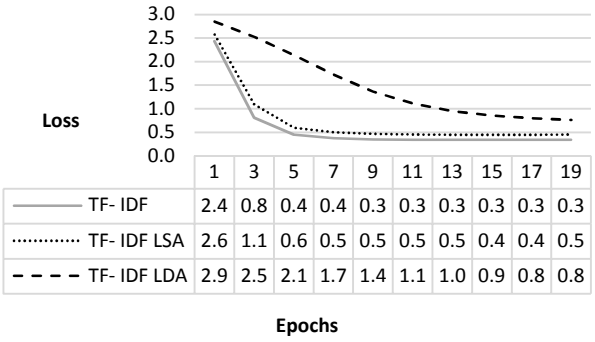
While feeding the classifier small datasets, the classifier performed with a very similar accuracy when two feature extraction approaches were used: TF-IDF and TF-IDF in combination with LSA. At the very beginning, the latter method outperformed the plain TF-IDF approach by enabling the classifier to achieve a higher validation accuracy by 1%. The TF-IDF enhanced with LDA approach did not allow the classifier to score high accuracy compared to other two methods when smaller datasets were fed (see Fig. 3).

When larger datasets were fed to the neural network, then the plain TF-IDF approach allowed the classifier to achieve higher validation accuracy compared to other feature extraction methods. This method enabled the classifier to achieve 91% accuracy with the largest dataset. Also, the feature extraction technique where TF-IDF was used in combination with LDA, enabled the classifier to rapidly increase validation accuracy when feeding the classifier larger datasets. By analyzing the training dataset, it was confirmed, that the TF-IDF enhanced with LDA feature extraction approach failed to reduce the noise in the data, thus resulting in the classifier having more difficulties to correctly classify text which falls under specific categories when working with smaller datasets.



	0.5	1	2	4	6	8	10
TF-IDF	72%	77%	88%	88%	89%	90%	91%
TF-IDF LSA	73%	77%	86%	87%	87%	87%	87%
TF-IDF LDA	55%	48%	46%	58%	76%	82%	83%

Fig. 3. Validation accuracy comparison.



	1	3	5	7	9	11	13	15	17	19
TF- IDF	2.4	0.8	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.3
TF- IDF LSA	2.6	1.1	0.6	0.5	0.5	0.5	0.5	0.4	0.4	0.5
TF- IDF LDA	2.9	2.5	2.1	1.7	1.4	1.1	1.0	0.9	0.8	0.8

Fig. 4. Validation loss comparison.

During neural network classifier training, the validation loss was analyzed as well. It was noticed, that when feeding to the classifier the data where features were extracted using TF-IDF enhanced with LDA method, the classifier gradually decreased the validation loss value after each epoch compared to other feature extraction methods (see Fig. 4).

Conclusions

This paper has compared in practice three different text feature extraction approaches by analyzing their influence over the accuracy of a neural network classifier. The TF-IDF feature extraction approach is found to outperform other methods allowing the classifier to achieve the highest validation accuracy score (91%) with the largest dataset.

What is more, while working with smaller datasets, the TF-IDF enhanced with LSA approach allows the neural network classifier to classify text just as good as using plain TF-IDF feature extraction.

Authorized licensed use limited to: University of Limerick. Downloaded on August 03,2025 at 14:02:57 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[2] C. C. Aggarwal (Ed.), "Data classification: algorithms and applications," CRC press, 2014.

[3] G. Diaz, *stopwords-lt*. Retrieved from Github: https://snowballstem.org/, 2016.

[4] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning," MIT press, 2016.

[5] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman and P. Blunsom, "Teaching machines to read and comprehend," in Advances in neural information processing systems, 2015, pp. 1693-1701.

[6] R. Maronna, "Alan julian izenman (2008): modern multivariate statistical techniques: regression, classification and manifold learning," Statistical Papers, vol. *52* no 3, pp. 733-734, 2011.

[7] J. Jiang, "Information extraction from text", in Mining text data, Springer, Boston, MA, 2012, pp. 11-41.

[8] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] A. Kolesau, D. Šešok and M. Rybokas, "A character-based part-of-speech tagger with feedforward neural networks", Romanian journal of information science and technology, vol 21, no. 4, pp. 446-459, 2018.

[10] T. K. Landauer, D. S. McNamara, S. Dennis and W. Kintsch (Eds.), Handbook of latent semantic analysis, Psychology Press, 2013.

[11] H. Liang, X. Sun, Y. Sun and Y. Gao, "Text feature extraction based on deep learning: a review," EURASIP journal on wireless communications and networking, vol. 2017, no 1, 211, 2017.

[12] C. D. Manning, C. D. Manning and H. Schütze, (1999). "Foundations of statistical natural language processing," MIT press, 1999, pp. 529-574.

[13] M. Porter, Retrieved from Snowball: https://snowballstem.org/, 2018.

[14] A. M. Rahman, A. Al Mamun and A. Islam, "Programming challenges of chatbot: Current and future prospective," in 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), IEEE, 2017, pp. 75–78.

[15] J. Ramos, "Using tf-idf to determine word relevance in document queries," in Proceedings of the first instructional conference on machine learning, vol. 242, 2003, pp. 133–142.

[16] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF," Journal of documentation, vol. 60, no. 5, pp. 503-520, 2004.

[17] A. M. Rush, S. Chopra and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[18] D. Shi, "A study on neural network language modeling," *arXiv preprint arXiv:1708.07252*, 2017.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The journal of machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, ... and J. Klingner, (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[21] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, ... and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in International conference on machine learning, 2015, pp. 2048–2057.

Authorized licensed use limited to: University of Limerick. Downloaded on August 03,2025 at 14:02:57 UTC from IEEE Xplore. Restrictions apply.