EXPLORING THE EFFECTIVENESS OF BERT USING DIFFERENT POOLING STRATEGIES ON SVM FOR NEWS CLASSIFICATION

SERENAY DOGANCA-CETIN

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE

DEPARTMENT OF
COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

STUDENT NUMBER

u966915

COMMITTEE

dr. Afra Alishahi
dr. Noortje Venhuizen

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

May 19, 2023

ACKNOWLEDGMENTS

The journey of working on this project has been an incredible learning experience for me. It has been a valuable and interdisciplinary challenge that has not only enhanced my coding skills, but also deepened my understanding of data science and research methodologies. Over the years, I have dedicated myself to studying the field, and this project has provided me with a profound insight into the techniques and principles I have been immersed in. Completing this project would not have been possible without the support and guidance of my supervisor Dr. Afra Alishahi, my friends, and my family. Their belief in my abilities has been instrumental in overcoming challenges and reaching this milestone. I am filled with a sense of accomplishment and appreciation as I present this thesis. I hope that the findings and insights presented within these pages contribute to the broader field of Cognitive Science and Artificial Intelligence.

EXPLORING THE EFFECTIVENESS OF BERT USING DIFFERENT POOLING STRATEGIES ON SVM FOR NEWS CLASSIFICATION

SERENAY DOGANCA-CETIN

Abstract

This study compares the performance of BERT-based approaches to the traditional TF-IDF method for news categorization task using the AG News Dataset. While previous studies have separately explored BERT embeddings and TF-IDF for text classification, we are directly comparing their performance on an SVM classifier for news categorization. The findings indicate that BERT embeddings can achieve competitive performance but are not consistently superior to TF-IDF representations. Previous studies have primarily focused on mean pooling or max pooling individually, but this study is extending this by exploring other pooling strategies. The study explores different feature extraction methods and demonstrates that mixed pooling outperforms the use of [CLS] tokens, and attention pooling may not consistently capture the most relevant information from embeddings, suggesting the need for alternative pooling methods. The research highlights that while BERT captures contextual information, it may not always be the optimal choice for every problem. TF-IDF representations, tailored to specific domains and vocabulary, prove effective for certain tasks and datasets. The study suggests that combining BERT embeddings with traditional methods, such as incorporating TF-IDF weights into the BERT attention mechanism, can enhance text classification and clustering models. Future work should focus on strategies for combining these techniques, including the exploration of domain-specific pre-trained BERT models and ensemble methods. In conclusion, the study highlights the importance of carefully selecting and evaluating text representation methods based on specific tasks and datasets, and suggests avenues for further improvement in text analysis models.

1 DATA SOURCE, ETHICS, CODE, AND TECHNOLOGY STATEMENT

The data used in this thesis is sourced from the AG News dataset, which is publicly available on Kaggle (Anand, n.d.). The owner of the data is the dataset provider, Aman Anand Rai. The dataset has been obtained from Kaggle following their terms of use and licensing agreement. All figures and images in this thesis have been created by the author unless explicitly mentioned otherwise. There are no figures or images in this thesis that require external consent.

The code used in this thesis is entirely developed by the author. No parts of the code have been borrowed or used from another study or external sources. In the process of writing this thesis, no tools or services were used to paraphrase the given text, check spelling or grammar, or typeset the text. The author manually conducted all writing, proofreading, and typesetting tasks without utilizing any specific tools or services.

2 INTRODUCTION

In recent years, the task of text classification has become increasingly important due to the vast amount of textual data generated in various domains. News classification is one such task, where the goal is to classify news articles into different categories based on their content. Traditional methods of text classification rely on hand-crafted features such as bag-of-words or Term Frequency-Inverse Document Frequency (TF-IDF). However, these methods may not capture the semantics and context of the text.

Recently, pre-trained language models such as Bidirectional Encoder Representations from Transformers (BERT) have shown promising results in various Natural Language Processing (NLP) tasks. BERT is a transformer-based model that learns contextualized embeddings of words in a sentence. These embeddings capture the contextual information of the words, which helps in improving the performance of downstream tasks such as text classification.

In this thesis, we explore the effectiveness of BERT embeddings for news classification on the AG News dataset. We compare the performance of Support Vector Machines (SVM) with BERT embeddings and TF-IDF as text representation methods. Different pooling methods for creating sentence-level embeddings are compared with the method of using [CLS] tokens as well. Unlike previous studies, we do not use a pre-trained BERT classifier but instead use BERT embeddings as text representation.

Our research questions are as follows:

RQ1 *Does using BERT embeddings as a text representation method improve the performance of SVM classifiers compared to TF-IDF?*

RQ2 *Which sentence representation as a result of feature extraction methods of mean-pooling, max-pooling, attention-pooling or mixed-pooling represents sentences better than [CLS] tokens?*

The scientific relevance of this study lies in exploring the effectiveness of using BERT embeddings as a text representation method for news classification. The proposed approach has the potential to improve the performance of text classifiers and can be applied to other NLP tasks as well. The rest of the thesis is organized as follows: Section 3 gives an overview of related work, Section 4 outlines the methodology used in the study, Section 5 presents the findings and analysis, and finally, Section 6 discusses the implications of the study's findings and suggests potential avenues for further research to address the limitations and explore new directions, and Section 7 provides a comprehensive conclusion and highlights the implications of the study's findings. It also offers recommendations for future research to further expand upon the current work.

The findings of this study reveal that the use of BERT embeddings as a text representation method in SVM classifiers for news classification does not consistently outperform TF-IDF. Additionally, the comparison of different pooling methods for sentence representation demonstrates that mixed pooling, which combines mean pooling and max pooling, outperforms the other pooling strategies and the use of [CLS] tokens.

3 RELATED WORK

The study by Barua, Sharif, and Hoque (2021) examines the six popular machine-learning techniques for Bengali sports news classification utilizing TF-IDF features. The highest performing algorithm and feature space combination were found to be the SVM with unigram+bigram+trigram with a weighted F1 score of 97.6%. Overall, SVM and NB obtained the best scores among the many combinations of feature space, and train and test data division rate. Kanika and Sangeeta (2019) discuss a news categorization system that uses TF-IDF and two supervised learning approaches, SVM and k-nearest neighbor (KNN) to categorize news articles from their titles and descriptions. They found that the use of the TF-IDF method enhanced both algorithms' efficiency as the SVM accuracy increased from 90% to 95% with the use of TF-IDF, and KNN accuracy increased from 53.33 to 96.66% (best case for k = 5). In the research carried out in 2021, Sunagar et al. implemented the classification of News Topic using AG's News Topic dataset using linear SVM, NB classifier, KNN, Rocchio, bagging, and boost-

ing. Before classification, tokenization, stop word removal, stemming, and TF-IDF was applied as pre-processing steps. The SVM had the greatest accuracy, at 91%, according to the results. These studies expose that in news categorization SVM performs the best when TF-IDF vectors are used.

TF-IDF has been traditionally employed for data representation. However, it is unable to take into account a word's placement or meaning within a sentence whereas the BERT model generates representations that consider the context and placement of words in sentences which has enabled its successful application of text classification. One of its key advantages is that it can be fine-tuned on specific tasks, such as sentiment analysis or text classification, by adding a simple classifier layer on top of the pre-trained model. Despite its impressive performance on these tasks, using BERT as a classifier model is not always the most time and cost-efficient approach. This is because fine-tuning a large language model like BERT can be computationally expensive, requiring powerful GPUs and large amounts of memory to train effectively. Furthermore, traditional machine learning models that use feature engineering and word embeddings can often achieve comparable results with much fewer computational resources (Gani & Chalaguine, 2022). Therefore, in this study, utilizing BERT embeddings with traditional machine learning algorithms is preferred over using BERT classifier models due to its greater time and cost efficiency.

The paper by Roman, Shahid, Uddin, Hua, and Maqsood (2021) explores the importance of citation intent in scientific publications, using BERT embeddings to represent text features. The performance of various machine learning classifiers was evaluated on two datasets, with the linear SVM achieving the best accuracy, particularly in an unbalanced dataset. Overall, SVM was found to be effective for text classification on top of contextual word embeddings. Their study supports our choice of using SVM classifiers with BERT embeddings as well. Subakti, Murfi, and Hariadi (2022) examined the effectiveness of BERT as a text representation in text clustering after applying various normalization and feature extraction methods to four different clustering algorithms. When obtaining BERT embeddings, max pooling and mean pooling feature extraction techniques were used. Their final results showed that BERT exceeded TF-IDF in 28 out of 36 metrics, with performance varying depending on the feature extraction method and normalization applied. Also, it was discovered that the scores in both KM and EFCM models were impacted by feature extraction and normalization techniques. We are adding more pooling techniques on top of mean-pooling and max-pooling which were used in their study, exploring other options in feature extraction.

Yu, Wang, and Jiang (2021) proposed an approach called BERT-BiGRU to address challenges in text classification such as metaphors, semantic

diversity, and grammar specificity. It does so by utilizing a BERT model for word representation in place of the conventional word2vec model, coupled with a BiGRU model which concurrently extracts text information features from both directions. The model was tested in Chinese text classification tasks and achieved outstanding performance; with accuracy, recall and F1 scores all above 90%. Nevertheless, there are some drawbacks, such as the need for larger computing power and duration for training the BERT model due to its large number of parameters. The paper by S, Sunagar, Rajarajeswari, and Kanavalli (2022) presents a hybrid model combining Long short-term memory (LSTM) and Gated recurrent unit (GRU) techniques to classify a Covid-19 Twitter dataset into 15 categories. The paper compares the performance of pre-trained word embedding techniques, GloVe and BERT, and finds that the BERT-hybrid model outperforms the GloVe-hybrid model, indicating contextual representation improves performance. These studies demonstrate the advantages of using BERT embeddings in capturing contextual information, handling linguistic complexities, and improving classification performance.

One study that resulted in a negative outcome for BERT embeddings is the study done by Vor Der Brück in 2020. In this study, the researchers analyzed the effectiveness of Bert embeddings in two NLP scenarios and compared them to Word2Vec embeddings. They used mean pooling and another approach that averages only over the start tokens that represent the beginning of a sentence to obtain sentence-level embeddings. As a result, the performance of Word2Vec embeddings was found to be considerably better in both scenarios and both sentence embedding methods.

According to Wang and Kuo (2020), different layers of BERT capture different linguistic properties, which can be leveraged to improve sentence representations. The authors propose a new sentence embedding method, called SBERT-WK, which dissects BERT-based word models through geometric analysis to find better sentence representations without further training. They evaluate SBERT-WK on various tasks and show that it achieves state-of-the-art performance. However, they have also found that the [CLS] token yields similar results in classification tasks. In our study, different pooling strategies will be utilized to form sentence representations and it will be discovered if they perform better than using [CLS] token. According to F. Chen, Datta, Kundu, and Beerel (2022), custom pooling strategies are ineffective since they only take into account the local context. They instead suggest a self-attentive pooling technique that can take the place of common pooling layers like max and average pooling. Multi-head self-attention, sigmoid activation, and exponential soft-max are the steps used in this self-attention module. During downsampling, their suggested approach effectively aggregates interdependence across non-local activa-

tion patches. The results of the trials demonstrate that, when used with different Convolutional Neural Network (CNN) architectures, this method performs better on object categorization and detection tasks than existing pooling strategies. Even though attention pooling was not used for BERT embeddings in their study, it offers insights into how well it performs. It was found by Jawahar and Sagot (2019) that lower layers in BERT tend to capture phrase-level information and surface features, while intermediate layers encode a rich hierarchy of linguistic information, starting with syntactic features in the middle and followed by semantic features at the top. That's why averaging the values across the first layers using mean pooling can help preserve the superficial characteristics. However, for upper layers, which capture more complex semantic and syntactic features, max pooling is more suitable which selects the most prominent features from these layers. In light of this information, applying mixed pooling which is the combination of mean pooling and max pooling on the different layers of BERT embeddings seems promising.

This study aims to address the research questions of whether using BERT embeddings as a text representation method can enhance the performance of SVM classifiers compared to TF-IDF and which pooling method out of mean pooling, max pooling, attention pooling, or mixed pooling, yields better results than using [CLS] tokens. The AG News dataset is used to conduct the experiments and evaluate the proposed approaches. By exploring the effectiveness of BERT embeddings in news classification tasks and comparing them to TF-IDF, this work fills the research gaps in understanding the potential improvements BERT can offer in SVM-based news classification. Additionally, the investigation of alternative sentence representation methods contributes to the existing knowledge by examining their performance in comparison to using [CLS] tokens. This study stands out from prior research by not relying on a pre-trained BERT classifier but leveraging BERT embeddings as text representation while employing traditional machine learning algorithms, and exploring different pooling methods. Our first baseline is the score of SVM classifier trained on TF-IDF vector, and the second baseline is the score of SVM classifier trained on [CLS] tokens.

4 METHOD

4.1 *Software*

Python was utilized to create numerous Jupyter notebooks for conducting experiments. These notebooks consisted of scripts specifically designed to carry out various tasks such as data extraction, analysis, cleaning, format-

	Class Index	Title	Description
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...
...	...	...	...
119995	1	Pakistan's Musharraf Says Won't Quit as Army C...	KARACHI (Reuters) - Pakistani President Perve...
119996	2	Renteria signing a top-shelf deal	Red Sox general manager Theo Epstein acknowled...
119997	2	Saban not going to Dolphins yet	The Miami Dolphins will put their courtship of...
119998	2	Today's NFL games	PITTSBURGH at NY GIANTS Time: 1:30 p.m. Line: ...
119999	2	Nets get Carter from Raptors	INDIANAPOLIS -- All-Star Vince Carter was trad...

120000 rows × 3 columns

Figure 1: Illustrative Data Representation

ting, preprocessing, applying machine learning algorithms, and visualizing the data. To handle the data efficiently, the pandas and NumPy libraries were employed, enabling the transformation of data into data frames and multidimensional arrays. During the data cleaning process, the re and string modules were utilized. To tokenize text data and generate contextualized word embeddings using the bert-base-uncased model, the Hugging Face Transformers library was employed. For feature extraction, the pretrained BERT transformer and TF-IDF was utilized. The torch library facilitated advanced computations on tensors. The Scikit-learn library was employed for preprocessing, hyperparameter tuning, classification, obtaining evaluation metrics, and visualizing confusion matrices.

4.2 *Dataset*

The dataset that was used in this study is the AG News Dataset which consists of over a million news articles. ComeToMyHead which is an academic news search engine and operational since July 2004 has gathered news articles from over 2000 news sources over the course of more than a year of operation. The dataset can be downloaded from this link (Anand, n.d.). AG News dataset is a good choice for a classification for containing news articles from various sources, its coverage of four categories: World, Sports, Business, and Sci/Tech, and its large size.

The AG News Dataset was created by selecting the four largest classes from the initial corpus. There are 1,900 testing samples and 30,000 training samples in each class, and 7,600 testing samples and 120,000 training samples overall. In both test and training datasets, the first column is "Class Id", the second column is "Title" and the third column is "Description". The "Class Id" column consists of numbers from 1 to 4 where 1 represents

"World", 2 represents "Sports", 3 represents "Business" and 4 represents "Sci/Tech". The distribution of the data among classes and the average number of words included in the "Title" and "Description" features are shown in Table 1.

	World	Sports	Business	Sci&Tech
Number of training samples	30000	30000	30000	30000
Number of test samples	1900	1900	1900	1900
The average number of words in the "Description" column	31	31	31	31
The average number of words in the "Title" column	7	6	7	7

Table 1: Information about the dataset among the classes.

4.3 *Preprocessing*

The preprocessing step is critical as it can significantly impact the performance of our machine-learning algorithms. By cleaning our text data, we can improve our classification accuracy and build more robust and accurate NLP models (Chai, 2022).

The first step in preprocessing our text data involves loading the data into a pandas DataFrame. We then concatenate the "Description" and "Title" columns of both the training and test set into a new column named "Combined". This concatenation allows us to capture more information from the text data and potentially improve our classification accuracy.

Next, we apply a series of text-cleaning techniques to remove unwanted characters and elements from our text data. We define a text cleaning function which uses regular expressions to remove HTML tags and special characters, such as & and {. Then, it creates a translation table using the string.punctuation and string.digits modules to remove all punctuation and digits from the text. Next, it uses regular expressions again to remove all non-alphabetic characters, such as numbers and special characters, from the text. Finally, it removes extra whitespace and converts the text to lowercase. The result is a cleaned version of the original text that contains only alphabetic characters in lowercase, with no punctuation or digits. We then apply this function to the "Combined" columns of our training and

test sets to remove any unwanted text elements. The cleaned data is used to create both BERT and TF-IDF representations.

After the cleaning process, the average number of words in data points belonging to each four classes is shown in Table 2.

	World	Sports	Business	Sci&Tech
The average number of words	37.83	36.36	36.36	36.28

Table 2: Information about the dataset among the classes after data cleaning.

4.4 *Feature extraction methods*

TF-IDF and BERT are two commonly used methods to obtain text representations. Both methods aim to transform raw text data into numerical vectors that can be used as input to machine learning models.

TF-IDF is a statistical method that is widely used in information retrieval and text mining. It measures the importance of a word in a document relative to the corpus of documents in which it appears. The method consists of two components: term frequency (TF) and inverse document frequency (IDF). Term frequency is a measure of how frequently a word appears in a document, while inverse document frequency measures how rare the word is across all documents in the corpus. By multiplying these two factors, we can get a score that reflects the importance of a word in a document. The TF-IDF scores are then normalized to make them comparable across documents. The resulting vector for a document represents the importance of each word in that document relative to the rest of the corpus (Z. Zhang, Lei, Xu, Mao, & Chang, 2019).

BERT, on the other hand, is a neural network-based language model that uses deep learning techniques to generate dense and context-aware word embeddings. In this study, the bert-base-uncased model was used to extract features which is a pre-trained model on the English language using a masked language modeling (MLM) objective. In the MLM process, a model randomly selects 15% of the words in a sentence and then predicts those words after processing the entire sentence (Devlin, Chang, Lee, & Toutanova, 2019). This model is uncased, which means that it does not make a difference between lower-case and upper-case letters.

BERT is a bidirectional model, meaning that it can capture the context of a word based on both it's preceding and succeeding words in a sentence. BERT can also handle tasks such as sentence classification, named entity recognition, and question answering. The resulting embeddings from

BERT are highly informative, as they capture not only the meaning of the individual words but also the relationship between them in a given context. Each transformer layer of 12-layer BERT (bert-base-uncased) creates a contextualized representation of each token by attending to different parts of the input sentence (Devlin et al., 2019).

For the bert-base-uncased model, the embeddings are created following the pseudo-code below:

for i in X_{train} **do**
    Tokenize input text with BERT's tokenizer;
    Convert tokens to tensor;
    Obtain BERT embeddings using the model;
end for

As a result, we obtain a *BaseModelOutputWithPoolingAndCrossAttentions* object for each data point in the input and it has the following attributes:

1. last_hidden_state: A torch.FloatTensor of shape (batch_size, sequence_length, hidden_size) containing the final hidden states of the last layer of the transformer for each token in the input sequence.

2. pooler_output: A torch.FloatTensor of shape (batch_size, hidden_size) containing the final hidden state of the [CLS] token after applying a linear transformation followed by a tanh activation function. The weights of this linear transformation are pretrained on the next sentence prediction task.

3. hidden_states: A tuple containing the hidden states for all layers (12 for the bert-base-uncased model) of the transformer. The shape of the hidden states for each layer is (batch_size, sequence_length, hidden_size).

4. past_key_values: A tuple containing the past key-value states that can be used for faster decoding. This attribute is only present if the model is a decoder (has the is_decoder attribute set to True).

5. cross_attentions: A tuple containing the cross-attention weights for all the cross-attention layers in the transformer. The shape of the cross-attention weights for each layer is (batch_size, num_heads, sequence_length, context_sequence_length).

4.5 *Feature Extraction - TF-IDF embeddings*

For the base case which uses TF-IDF as a text representation method, we applied TF-IDF vectorization to represent the preprocessed text data numerically. This technique assigns weights to the terms in the text based on their frequency in the current document and their rarity across all documents in the dataset. The fit_transform method fits the vectorizer on the training data and then transforms it into a matrix of TF-IDF vectors. The transform method is then used on the test data to transform it into TF-IDF vectors based on the fitted vectorizer from the training data. The resulting matrices have the same number of rows as the training and test data, respectively, and the number of columns is equal to the number of unique words in the training set which is 91240. The resulting TF-IDF matrix is a numerical representation of the text data that can be used for training the machine learning algorithms.

4.6 *Feature Extraction - BERT embeddings*

After cleaning our text data, we load the BERT model and tokenizer using the AutoModel and AutoTokenizer functions from the Hugging Face Transformers library to create BERT embeddings. We define a function that leverages the BERT model and tokenizer to generate sentence-level embeddings for each text data point in our dataset. This function tokenizes the input with the specified settings for adding special tokens, padding, and maximum length. The *maximum length* is calculated as 217 which is the length of the longest data point in the data set. The padding parameter is set to this maximum length, which means that any sequences that are shorter than the maximum length will be padded with special tokens to make them that length. The resulting embeddings belonging to each row in the dataset have shape *(sequence_length, hidden_size)* where *sequence_length* equals to *maximum length* which is 217 and hidden_size equals to 768. These embeddings allow us to represent the text data in a dense and continuous vector space, which is an essential step for any machine learning algorithm (*Embeddings | Machine Learning Crash Course | Google Developers*, 2019),

The embeddings acquired are converted to sentence-level embeddings because sentence embeddings provide a fixed-size representation of variable-length sentences, which is required for training machine learning classifiers.

To obtain sentence-level embeddings, it is generally recommended to use the [CLS] token from BERT's output (Choi, Kim, Joe, & Gwon, 2021). The [CLS] token is a special token used in the BERT model and is located at the beginning of the input sequence. It stands for "classification" and is used to represent the input sequence as a whole for tasks such as sentence

classification or sentiment analysis. The [CLS] token is added by the BERT model during the input preprocessing step and is included in the final output representations of the model. In the original BERT model, the [CLS] token is used for next-sentence prediction loss, serving as a sentence embedding. So, the first approach to obtain sentence-level embeddings in this study is to use the hidden states of the [CLS] token of the last layer as inputs to the machine learning algorithms and use it as a base case when comparing different sentence-level representations.

To generate sentence-level embeddings, we divide the data into batches with a size of 512 to fit into the memory of our machine. Then for each batch, the BERT embeddings of each input text are computed and one of the five selected methods is performed to obtain a single sentence embedding. The size of each embedding is 768, and the number of embeddings is equal to the number of input texts in the training and test set which are 120.000 and 7600 respectively.

The BERT model used in this study was "bert-base-uncased", which is a pre-trained model that has been trained on a large corpus of text. The model was loaded using the PyTorch library and the Hugging Face Transformers library. Five strategies were used to aggregate the word embeddings into a sentence-level representation: using the [CLS] token, mean-pooling, max-pooling, attention-pooling, and mixed-pooling (mean & max). As a result of the pooling process, we obtain an array with a size of 768 instead of (217, 768).

The mean, max, and attention poolings are applied to the final hidden states of the last layer of the transformer. On the other hand, the mixed pooling method utilizes all the layers in the output.

1. The mean pooling technique takes the average of all the word embeddings to obtain a single sentence embedding, except for the padded tokens. As a result, we obtain a single vector of length 768 that represents the mean of all 217 vectors. Mean pooling is useful when we want to capture the overall sentiment or tone of the text data (Zhao, You, Chang, Zhang, & Hu, 2022).

2. The max pooling technique selects the maximum value from each dimension (column) of the word embeddings to obtain a single sentence embedding. As a result, we obtain a single vector of length 768 that represents the maximum of all 217 vectors. Max pooling is useful when we want to capture the most important information from the text data (Lehečka, Švec, Ircing, & Šmídl, 2020).

3. Attention pooling incorporates attention mechanisms to weigh the importance of each token when aggregating the embeddings. The

attention pooling technique generates sentence-level embeddings by giving more attention to important words in the text data (Trinh, Luong, & Le, 2019). The steps to create sentence-level embeddings using attention pooling are:

(a) First, we obtain the BERT embeddings for the input sentence.
(b) The attention weights are computed by applying a softmax activation on the output of the attention layer.
(c) Attention pooling is then performed by element-wise multiplication of the hidden states with the attention weights, followed by summing along the sequence length dimension.

$$\begin{aligned} \text{BERT}_{\text{attention}}(s) &= \text{sum}\left(\{h_1, h_2, \ldots, h_n\} \cdot \{a_1, a_2, \ldots, a_n\}\right) \\ &\text{where:} \\ h_i &= \text{BERT}(w_i), \quad w_i \in \text{s} \\ a_i &= \text{softmax}(w_i \cdot w_n), \quad w_i \in \text{s} \end{aligned} \tag{1}$$

4. Different layers of BERT capture different linguistic properties, and fusing information across layers can result in better sentence representations for classification tasks (Wang & Kuo, 2020). The mixed pooling that was used in this study first obtains the BERT embeddings for the input text, and then performs mixed pooling by taking the average pooling of the first six layers, and max pooling of the remaining six layers as 0.5 was found to be the most optimum ratio of mean and max pooling when applying mixed pooling by Li et al. (2021). Finally, the embeddings from both layers are concatenated to obtain the final sentence embedding. The embedding size is also updated from 768 to 1536, as the mixed pooling method generates embeddings of size 2*768=1536.

$$\begin{aligned} \text{BERT}_{\text{mixed}}(s) &= \text{concat}\left(\text{mean\_pool}\left(\{h_{1:6}\}\right), \text{max\_pool}\left(\{h_{7:n}\}\right)\right) \\ &\text{where:} \quad h_i = \text{BERT}(w_i), \quad w_i \in s \end{aligned} \tag{2}$$

After generating sentence-level embeddings for the training set, we can use these embeddings to train a machine-learning model to classify our text data.

4.7 *Machine Learning Algorithm*

SVMs are a type of machine learning classifier that uses kernel functions to transform the input data into a higher-dimensional space where it can be

more easily separated by a linear decision boundary, introduced by Cortes and Vapnik (1995). The kernel trick is used to map the data into a higher-dimensional space before solving the machine-learning task. The most popular kernel functions used in SVMs are linear, polynomial, Gaussian (RBF), and sigmoid. The polynomial kernel creates a non-linear decision boundary by mapping the original dataset into a higher dimensional space. The choice of kernel function depends on the nature of the data and the problem being solved (Hechter, 2004). The SVM training algorithm generates a model that categorizes new instances into one of two groups, functioning as a deterministic binary linear classifier. This is accomplished by undergoing a series of training procedures. SVM operates by identifying the optimal hyperplane that effectively divides the data into distinct classes. The selection of the hyperplane is based on maximizing the separation margin between the two classes. SVM can also be used for multiclass classification by using a one-vs-rest (OVR) or one-vs-one (OVO) approach (Saxena, Anamika, Pant, & Tripathi, 2019). SVMs have gained popularity in data mining, pattern recognition, and machine learning communities due to their optimal solution, discriminative power, and extraordinary generalization capability. SVMs have been shown to be superior to other supervised learning methods and have become one of the most used classification methods. The decision functions are determined directly from the training data by maximizing the separation between decision borders in a high-dimensional space called the feature space. This classification strategy minimizes classification errors and obtains better generalization ability (Cervantes, Garcia-Lamont, Rodríguez-Mazahua, & Lopez, 2020).

4.8 *Experimental Setup*

After obtaining numerical vector representations from BERT and TF-IDF, they are fed into the SVM classifier. A parameter grid is defined to search for hyperparameter tuning. The parameter grid contains different values of 'C', 'kernel', and 'degree' which can be seen below.

```
param_grid = {
'C': [0.1, 1, 10],
'kernel': ['linear', 'rbf', 'poly'],
'degree': [2, 3, 4]
}
```

Due to the computational complexity and memory requirements of the SVM algorithm, the training data was divided into batches with a size of 1000. The code then loops through the training data in batches, retrieves each batch's data, and performs a grid search using three-fold cross-validation to find the best hyperparameters for that batch. The best hyperparameters are then added to the list. After all batches have been processed, the code computes the most common

hyperparameters among all batches and creates a new SVM classifier using those hyperparameters. Finally, the entire training data is fed into the new classifier, and the accuracy score and F1 score are computed on the test data. The accuracy score measures the proportion of correctly classified instances, while the F1-score is the harmonic mean of the precision and recall. When calculating F1 scores, macro averaging was used since each class has the same number of samples in our dataset and macro-averaged F1 score is computed as the arithmetic mean of all the per-class F1 scores, treating all classes equally (Leung, 2022).

This process is repeated for every BERT embedding as a result of different poolings and TF-IDF vectors. The best hyperparameters for each representation are available in Table 3.

	C	degree	kernel
BERT - [CLS]	1	2	linear
BERT - Mean-pooling	1	4	poly
BERT - Max-pooling	1	2	linear
BERT - Attention pooling	1	4	poly
BERT - Mixed pooling	1	3	poly
TF-IDF	1	2	linear

Table 3: The optimum hyperparameters for SVM classifier using different individual text representations.

4.9 *Evaluation Metrics*

Accuracy and F1 score are two common evaluation metrics used in machine learning and natural language processing tasks to assess the performance of models. Accuracy is the ratio of correctly predicted instances to the total instances in the dataset. It is calculated as:

$$\text{Accuracy} = \frac{\textit{True Positives} + \textit{True Negatives}}{\textit{True Positives} + \textit{True Negatives} + \textit{False Positives} + \textit{False Negatives}} \tag{3}$$

Accuracy is a widely used metric, but it may not be suitable for imbalanced datasets, where one class significantly outnumbers the other classes (H. Zhang et al., 2019). F1 score, on the other hand, is the harmonic mean of precision and recall. Precision is the ratio of True Positives to the sum of True Positives and False Positives, while recall is the ratio of True Positives to the sum of True Positives and False Negatives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4}$$

$$\text{Recall} = \frac{\textit{TruePositives}}{\textit{TruePositives} + \textit{FalseNegatives}} \tag{5}$$

The F1 score is calculated as follows:

$$\text{F1} == \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{6}$$

It ranges from 0 to 1, with 1 being the best possible score. F1 score is particularly useful when dealing with imbalanced datasets, as it takes both false positives and false negatives into account, providing a more balanced evaluation of the model's performance (Liu, Yao, Zhou, Wang, & Huang, 2023).

5 RESULTS

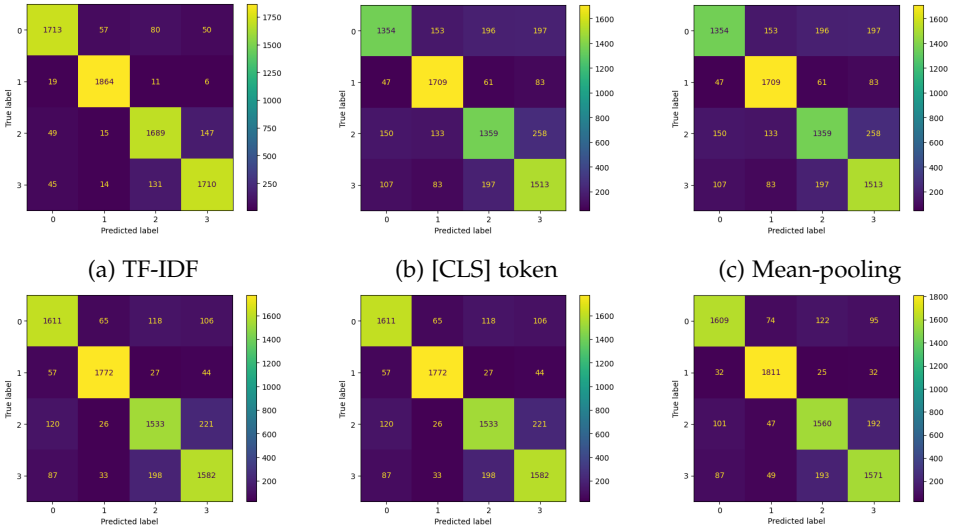
The accuracy and F1 scores of SVM classifiers trained on different text representations and the optimum hyperparameters are available in Table 4 and the confusion matrices can be seen in Figure 2.

	Accuracy	F1 score
TF-IDF	0.918	0.918
BERT - [CLS]	0.807	0.807
BERT - Mean-pooling	0.781	0.780
BERT - Max-pooling	0.855	0.855
BERT - Attention pooling	0.780	0.779
BERT - Mixed pooling	0.862	0.862

Table 4: The accuracy and F1 scores of SVM classifiers trained on different individual text representations.

Based on the results, we can observe the accuracy and F1 scores of SVM classifiers trained on different word embeddings, namely BERT with different pooling strategies (BERT - [CLS], BERT - mean-pooling, BERT - max-pooling, BERT - attention pooling, and BERT - mixed pooling), as well as TF-IDF.

1. TF-IDF (First baseline): The SVM classifier trained on TF-IDF representations achieves an accuracy of 0.918 and an F1 score of 0.918. The accuracy score of 0.918 indicates that the SVM classifier performs quite well when using TF-IDF vectors as features. TF-IDF is a commonly used text representation method that captures the importance of terms in documents, and the results suggest that it is effective for this classification task.
2. BERT - [CLS] (Second baseline): The SVM classifier trained on BERT embeddings using the [CLS] token achieves an accuracy of 0.807 and an F1 score of 0.807. This approach involves taking the embedding of the [CLS] token, which represents the entire input sequence, as the fixed-dimensional representation for classification. The accuracy and F1 scores indicate that this approach performs reasonably well.
3. BERT - Mean-pooling: The SVM classifier trained on BERT embeddings using mean-pooling achieves an accuracy of 0.781 and an F1 score of 0.780.



(a) TF-IDF (b) [CLS] token (c) Mean-pooling

(d) Max-pooling (e) Attention-pooling (f) Mixed-pooling

Figure 2: Confusion matrices

Mean-pooling involves taking the average of all token embeddings in the input sequence. Comparing it with the [CLS] token approach, mean-pooling seems to perform slightly worse in terms of accuracy and F1 score.

4. BERT - Max-pooling: The SVM classifier trained on BERT embeddings using max-pooling achieves an accuracy of 0.855 and an F1 score of 0.855. Max-pooling involves taking the maximum value across each dimension of the token embeddings. This approach performs better than both BERT - [CLS] and BERT - Mean-pooling in terms of accuracy and F1 score.

5. BERT - Attention pooling: The SVM classifier trained on BERT embeddings using attention pooling achieves an accuracy of 0.780 and an F1 score of 0.779. Attention pooling involves calculating attention weights for each token and using them to compute a weighted sum of token embeddings. It seems to perform similarly to the BERT - Mean-pooling approach but slightly worse than BERT - [CLS] and BERT - Max-pooling.

6. BERT - Mixed pooling: The SVM classifier trained on BERT embeddings using mixed pooling achieves an accuracy of 0.862 and an F1 score of 0.862. Mixed pooling combines both mean-pooling and max-pooling by concatenating the two representations. This approach achieves the highest accuracy and F1 score among the BERT-based approaches, indicating its effectiveness.

Overall, based on the provided results, the SVM classifier trained on TF-IDF vectors outperforms all the BERT-based approaches in terms of accuracy and F1-score.

6 DISCUSSION

The goal of this study was to compare the performance of BERT embeddings and TF-IDF representations in text classification tasks and explore the effect of using different sentence-level embeddings for BERT. Our findings shed light on the strengths and limitations of these text representation methods, highlighting the need for careful consideration when choosing the appropriate method for specific problems and datasets.

First, we observed that BERT embeddings can provide competitive performance in text classification tasks when compared to traditional methods such as TF-IDF. However, the performance of BERT embeddings is not universally superior, as evidenced by the results on the AG News, Yahoo! Answers, and R2 datasets (Subakti et al., 2022). In some cases, TF-IDF representations outperformed BERT embeddings, indicating that the choice of text representation method should be carefully considered based on the specific problem and dataset at hand.

One possible explanation for the varying performance is the different nature of the two methods. BERT is a powerful model that captures contextual information in text, but it may not always be the best choice for every problem. BERT is pre-trained on a large corpus from various sources, which is not as specific to the AG News dataset as the TF-IDF representation. On the other hand, TF-IDF is calculated directly from the dataset and it focuses on the frequency of words in documents and their importance across a collection of documents, making it more tailored to the specific domain and vocabulary used in the news articles.

Additionally, the high dimensionality of BERT embeddings can pose challenges for clustering or classification tasks. While BERT captures fine-grained contextual information, its embeddings have a large number of dimensions. This can lead to difficulties in effectively leveraging the information contained in the embeddings for certain algorithms or tasks. In contrast, TF-IDF vectors are sparse and have lower dimensionality, which might make them more suitable for certain algorithms.

Furthermore, our results indicate that the choice of sentence representation method within BERT embeddings can also impact performance. Among the different feature extraction methods, the mixed pooling approach achieved the best performance, outperforming both the [CLS] token and other pooling strategies. This suggests that a combination of mean-pooling and max-pooling, capturing both average and maximum information from the embeddings, can result in more robust representations for classification tasks. We also observed that attention pooling performed poorly. Attention pooling is designed to weigh the importance of different words in a sentence, but it may not always identify the most critical features for a specific task (Rasmy, Xiang, Xie, Tao, & Zhi, 2021). This highlights the importance of carefully considering the pooling strategy when using BERT embeddings and suggests that other pooling approaches, such as mixed pooling, may be more effective for certain text classification tasks.

It is important to note that our study has some limitations. Firstly, we evaluated the performance of BERT embeddings and TF-IDF representations on a specific dataset, and the results may not generalize to other domains or datasets. Future studies should consider evaluating these methods on a wider range of datasets to further validate our findings. Additionally, our study focused on the

performance of SVM classifiers using BERT embeddings and TF-IDF representations. Other classification algorithms or models may yield different results, and exploring the performance of BERT embeddings with different classifiers would provide a more comprehensive analysis. Furthermore, our study did not consider fine-tuning BERT on the specific task at hand. Investigating the impact of fine-tuning BERT on text classification could provide valuable insights into the potential benefits and trade-offs of incorporating task-specific information.

7 CONCLUSION

Regarding the research questions:

1. Does using BERT embeddings as a text representation method improve the performance of SVM classifiers compared to TF-IDF?

   In this study, the SVM classifier trained on TF-IDF representations outperforms all the BERT-based methods, achieving an accuracy of 0.918. The BERT-based methods have lower accuracy scores, with mixed pooling achieving the highest accuracy of 0.862 among them.

2. Which sentence representation as a result of feature extraction methods of mean-pooling, max-pooling, attention-pooling, or mixed-pooling represents sentences better than [CLS] tokens?

   Based on the results, the mixed pooling method achieves the best performance among the BERT-based methods, with an accuracy of 0.862 and an F1 score of 0.862. This outperforms the [CLS] token method, which has an accuracy of 0.807 and an F1 score of 0.807.

In conclusion, our study demonstrates that the performance of BERT is not universally superior, and TF-IDF representations can outperform BERT embeddings in certain cases. The choice of text representation method should be carefully considered based on the specific problem and dataset. Incorporating TF-IDF weighting into the BERT model has been shown to enhance performance in some cases (W. Chen et al., 2020), suggesting that combining the strengths of both BERT embeddings and traditional text representation methods can lead to more robust and accurate models for text classification and clustering tasks.

In future work, we recommend exploring other strategies for combining BERT embeddings with traditional text representation methods, such as incorporating TF-IDF weights into the BERT attention mechanism. Furthermore, investigating the use of domain-specific pre-trained BERT models and ensemble methods could potentially lead to further improvements in performance. Future research should address our limitations as well by considering a wider range of datasets, exploring different classification algorithms, investigating the impact of fine-tuning, and utilizing a more comprehensive set of evaluation metrics. By doing so, we can gain a deeper understanding of the strengths, weaknesses, and optimal use cases of BERT embeddings and TF-IDF representations in various text-related applications.

REFERENCES

Anand, A. (n.d.). *Ag news classification dataset.* Retrieved from https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset

Barua, A., Sharif, O., & Hoque, M. M. (2021). Multi-class sports news categorization using machine learning techniques: Resource creation and evaluation. *Procedia Computer Science*, *193*, 112-121. doi: 10.1016/j.procs.2021.11.002

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020, 09). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, *408*, 189–215. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231220307153 doi: 10.1016/j.neucom.2019.10.118

Chai, C. P. (2022, 06). Comparison of text preprocessing methods. *Natural Language Engineering*, 1-45. doi: 10.1017/s1351324922000213

Chen, F., Datta, G., Kundu, S., & Beerel, P. (2022, 12). *Self-attentive pooling for efficient deep learning.* Retrieved 2023-05-11, from https://arxiv.org/pdf/2209.07659.pdf

Chen, W., Yuan, X., Zhang, S., Wu, J., Zhang, Y., & Wang, Y. (2020). *Ferryman at semeval-2020 task 3: Bert with tfidf-weighting for predicting the effect of context in word similarity.*

Choi, H., Kim, J., Joe, S., & Gwon, Y. (2021, 01). *Evaluation of bert and albert sentence embedding performance on downstream nlp tasks.* Retrieved from https://arxiv.org/pdf/2101.10642.pdf

Cortes, C., & Vapnik, V. (1995, 09). Support-vector networks. *Machine Learning*, *20*, 273-297. doi: 10.1007/bf00994018

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, 05). *Bert: Pre-training of deep bidirectional transformers for language understanding.*

Embeddings | machine learning crash course | google developers. (2019). Retrieved from https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

Gani, R., & Chalaguine, L. (2022, 10). *Feature engineering vs bert on twitter data.* Retrieved 2023-05-12, from https://arxiv.org/ftp/arxiv/papers/2210/2210.16168.pdf

Hechter, T. (2004). *A comparison of support vector machines and traditional techniques for statistical regression and classification*. 53-59.

Jawahar, G., & Sagot, B. (2019). *What does bert learn about the structure of language?* Retrieved 2023-05-09, from https://aclanthology.org/P19-1356.pdf

Kanika, & Sangeeta. (2019). Applying machine learning algorithms for news articles categorization: Using svm and knn with tf-idf approach.

Smart Computational Strategies: Theoretical and Practical Aspects, 95-105. doi: 10.1007/978-981-13-6295-8_9

Lehečka, J., Švec, J., Ircing, P., & Šmídl, L. (2020). Adjusting bert's pooling layer for large-scale multi-label text classification. *Text, Speech, and Dialogue*, *12284*, 214-221. doi: 10.1007/978-3-030-58323-1_23

Leung, K. (2022, 01). *Micro, macro weighted averages of f1 score, clearly explained.* Retrieved from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Li, J., Zhang, Z., Chen, M., Ma, J., Wang, S., & Xiao, J. (2021, 08). Improving polyphone disambiguation for mandarin chinese by combining mix-pooling strategy and window-based attention. *Interspeech 2021*. doi: 10.21437/interspeech.2021-1232

Liu, J., Yao, J., Zhou, Q., Wang, Z., & Huang, L. (2023, 04). Lstmae-dwsslm: A unified approach for imbalanced time series data classification. *Applied Intelligence*, *1*, 3. doi: 10.1007/s10489-023-04642-0

Rasmy, L., Xiang, Y., Xie, Z., Tao, C., & Zhi, D. (2021, 05). Med-bert: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *npj Digital Medicine*, *4*. doi: 10.1038/s41746-021-00455-y

Roman, M., Shahid, A., Uddin, M. I., Hua, Q., & Maqsood, S. (2021, 04). Exploiting contextual word embedding of authorship and title of articles for discovering citation intent classification. *Complexity*, *2021*, 1-13. doi: 10.1155/2021/5554874

S, S., Sunagar, P., Rajarajeswari, S., & Kanavalli, A. (2022). Bert-based hybrid rnn model for multi-class text classification to study the effect of pre-trained word embeddings. *International Journal of Advanced Computer Science and Applications*, *13*. doi: 10.14569/ijacsa.2022.0130979

Saxena, A., Anamika, Pant, B., & Tripathi, V. (2019, 12). Text mining for multiclass research paper categorization. *International Journal of Innovative Technology and Exploring Engineering*, *9*, 2612-2615. doi: 10.35940/ijitee.b7240.129219

Subakti, A., Murfi, H., & Hariadi, N. (2022, 02). The performance of bert as data representation of text clustering. *Journal of Big Data*, *9*. doi: 10.1186/s40537-022-00564-9

Sunagar, P., Kanavalli, A., Nayak, S. S., Mahan, S. R., Prasad, S., & Prasad, S. (2021). News topic classification using machine learning techniques. *Lecture Notes in Electrical Engineering*, *733*, 461-474. doi: 10.1007/978-981-33-4909-4_35

Trinh, T., Luong, M.-T., & Le, Q. (2019, 07). *Selfie: Self-supervised pretraining for image embedding.*

Vor Der Brück, T. (2020). *Employing bert embeddings for customer segmentation and translation matching.*

Wang, B., & Kuo, C.-C. J. (2020, 06). Sbert-wk: A sentence embedding method by dissecting bert-based word models. , *14*. Retrieved 2023-05-09, from https://arxiv.org/pdf/2002.06652.pdf

Yu, Q., Wang, Z., & Jiang, K. (2021, 01). Research on text classification based on bert-bigru model. *Journal of Physics: Conference Series*, *1746*, 012019. doi: 10.1088/1742-6596/1746/1/012019

Zhang, H., Li, Z., Shahriar, H., Tao, L., Bhattacharya, P., & Qian, Y. (2019, 07). Improving prediction accuracy for logistic regression on imbalanced datasets. *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. doi: 10.1109/compsac.2019.00140

Zhang, Z., Lei, Y., Xu, J., Mao, X., & Chang, X. (2019, 09). Tfidf-fl: Localizing faults using term frequency-inverse document frequency and deep learning. *IEICE Transactions on Information and Systems*, *E102.D*, 1860-1864. doi: 10.1587/transinf.2018edl8237

Zhao, S., You, F., Chang, W., Zhang, T., & Hu, M. (2022, 02). Augment bert with average pooling layer for chinese summary generation. *Journal of Intelligent Fuzzy Systems*, *42*, 1859-1868. doi: 10.3233/jifs-211229