emerald PUBLISHING

Data Technologies & Appli

Harmonized System Code Classification using Supervised Contrastive Learning with Sentence BERT and Multiple Negative Ranking Loss

Journal:	*Data Technologies and Applications*
Manuscript ID	DTA-01-2024-0052.R1
Manuscript Type:	Article
Keywords:	Sentence BERT, Multiple Negative Ranking Loss, Harmonised System Code, Trade Transactions, Support Vector Machine, Random Forest

Harmonized System Code Classification using Supervised Contrastive Learning with Sentence BERT and Multiple Negative Ranking Loss

Abstract

Purpose — International trade transactions, extracted from customs declarations, include several fields, among which the product description and the product category are the most important. The product category, also referred to as the Harmonised System Code (HS code), serves as a pivotal component for determining tax rates and administrative purposes. A predictive tool designed for product categories or HS codes becomes an important resource aiding traders in their decision to choose a suitable code. This tool is instrumental in preventing misclassification arising from the ambiguities present in product nomenclature, thus mitigating the challenges associated with code interpretation. Moreover, deploying this tool would streamline the validation process for government officers dealing with extensive transactions, optimising their workload and enhancing tax revenue collection within this domain.

Design/methodology/approach — This study introduces a methodology focused on the generation of sentence embeddings for trade transactions, employing Sentence BERT (SBERT) framework in conjunction with the Multiple Negative Ranking (MNR) Loss function following a contrastive learning paradigm. The procedure involves the construction of pairwise samples, including anchors and positive transactions. The proposed method is evaluated using two publicly available real-world datasets, specifically the India Import 2016 and United States Import 2018 datasets, to fine-tune the SBERT model. Several configurations involving pooling strategies, loss functions, and training parameters are explored within the experimental setup. The acquired representations serve as inputs for traditional machine learning algorithms employed in predicting the product categories within trade transactions.

Findings — Encoding trade transactions utilising SBERT with MNR loss facilitates the creation of enhanced embeddings that exhibit improved representational capacity. These fixed-length embeddings serve as adaptable inputs for training machine learning models, including Support Vector Machine (SVM) and Random Forest, intended for downstream tasks of HS code classification. Empirical evidence supports the superior performance of our proposed approach compared to fine-tuning transformer-based models in the domain of trade transaction classification.

Originality/value — Our approach generates more representative sentence embedding by creating the networks architectures from scratch with the SBERT framework. Instead of exploiting a data augmentation method generally used in contrastive learning for measuring the similarity between the samples, we arranged positive samples following a supervised paradigm and determined loss through distance learning metrics. This process involves continuous updating of the Siamese or bi-encoder network to produce embeddings derived from commodity transactions. This strategy aims to ensure that similar concepts of transactions within the same class converge closer within the feature embedding space, thereby improving the performance of downstream tasks.

Keywords — Sentence BERT, Multiple Negative Ranking Loss, Harmonised System Code, Trade Transactions, Support Vector Machine, Random Forest

Paper type — Research paper

1. Introduction

International trade, a fundamental component of the global economy, encompasses the exchange of goods or commodities across national borders through import and export activities. The continuous evolution of logistics technology and the expansion of e-commerce have notably amplified the volume of trade transactions over time. As a consequence, customs duties and taxes, pivotal components of a country's fiscal structure, contribute considerably to its overall revenue. Statistics from the World Customs Organization (World Customs Organization, 2022) suggest that, on average, 25.2% of a nation's tax revenue is derived from these customs duties and taxes, demonstrating the substantial impact of international trade on national economies.

The governmental role in governing international trade involves the establishment of tariffs, the regulation of import and export permissions for various commodities, and the negotiation and facilitation of trade agreements among nations. In facilitating international trade, traders are mandated to submit a customs declaration containing comprehensive details about the traded goods. This declaration comprises a comprehensive portrayal of the commodity alongside the allocation of a Harmonised System code (HS code) that identifies its categorisation, as prescribed by the (World Customs Organization, 2013). HS codes serve a pivotal role in ascertaining the applicable tax rates for products and fulfilling essential administrative functions within international trade frameworks. However, the process of selecting an appropriate HS code can be challenging due to the ambiguities contained in the textual descriptions within the product nomenclature system, making precise identification a complex task.

Machine learning offers a viable solution for predicting the categorical classification of products based on their descriptions. This predictive capability is useful for traders to prevent unintentional misclassification of products, which could result in penalties or fines. Moreover, with the escalating volume of transactions, regulatory authorities strive to ensure traders' adherence to procedural protocols during product exchanges. The validation of these transactions plays an essential role in verifying traders' compliance with tax obligations, thereby safeguarding national revenues.

Integrating automated technological support is imperative in this context. It prevents the introduction of subjective judgement, often influenced by the varying experiences of different officers, ensuring unbiased decision-making. Simultaneously, it ensures the timely provision of services without disrupting the trading processes, thereby bolstering the efficiency and integrity of international trade operations.

The prediction of HS codes, leveraging product description declarations, necessitates the conversion of textual commodity data into numerical representations to facilitate machine learning classification tasks. The field of natural language processing (NLP) has witnessed substantial evolution in recent decades. Within this progression, considerable focus has been devoted to transforming sequences into numerical structures, manifesting in both sparse and dense representations of language.

The advent of Large Language Models (LLMs), such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin *et al.*, 2019) marked a pivotal milestone in NLP. Built upon the transformer model architecture with attention mechanisms (Vaswani *et al.*, 2017), BERT introduced a revolutionary paradigm in textual representation. Trained on vast corpora, BERT models exhibit robust efficacy in encapsulating textual information and exhibiting exceptional performance in various language-related tasks. Its distilled variant, known as DistilBERT (Sanh *et al.*, 2020), accomplishes a 40% reduction in size compared to BERT while retaining an impressive 97% of its language understanding capabilities and enhancing operational speed by 60%. Prior research endeavours have involved fine-tuning a DistilBERT model using a dataset comprising product descriptions from trade transactions.

These efforts have showcased promising outcomes in predicting HS codes for previously unseen transactions (Anggoro *et al.*, 2023). This empirical evidence underlines the potential efficacy of employing DistilBERT in HS code prediction tasks within the realm of customs declaration processing.

This study represents an extension and enhancement of the prior work laid by (Anggoro *et al.*, 2023) by investigating the generation of sentence embeddings tailored for trade transactions, achieved through the customisation of the SBERT framework's foundational components. Diverging from previous methodologies focused on fine-tuning transformer-based models, this research focuses on utilising the SBERT framework (Reimers and Gurevych, 2019). SBERT adopts siamese or bi-encoder network architecture, effectively producing more meaningful and representative embeddings. These embeddings are subsequently leveraged in a downstream task, specifically HS code prediction.

The methodology proposed in this research is rigorously assessed by utilising two publicly available datasets that include international trade transactions originating from diverse countries. The empirical findings distinctly illustrate the efficacy of generating sentence embeddings with the SBERT model and subsequently employing traditional machine learning algorithms for text classification. As a result, this approach surpasses the performance achieved by fine-tuning transformer-based models. The results explicitly highlight the pivotal role of enhanced transaction representation in improving model performance in HS code prediction tasks.

2. Related Works

2.1. Classification Task for Commodity Trade

Advancements within the research field of NLP have flourished over recent decades, consequently enhancing model performance within the domain of text classification (Kowsari *et al.*, 2019). Classification tasks have found extensive applications as downstream tasks, aiming to categorise textual data into specific classes. Within the domain of international trade, product descriptions serve as the input data, while the product category or HS code stands as the designated labels. Both elements are imperative for training the model within the supervised learning paradigm, providing knowledge to the model to enable subsequent predictions of product categories or HS codes based on the product descriptions.

Numerous approaches have been explored to facilitate classification tasks, commencing with conventional methods like fuzzy logic (Singh and Sahu, 2004), which are characterised by their simplicity and limited adaptability. Such systems tend to rely heavily on personal knowledge and demonstrate limitations in addressing intricate issues, often resulting in divergent assessments and biassed justifications.

Traditional machine learning classifiers, such as SVM and Random Forest, have exhibited notable efficacy (Altaheri and Shaalan, 2020). In this particular study, Term Frequency-Inverse Document Frequency (TF-IDF) is employed as a feature to portray transaction content. This technique yields sparsely represented features, disregarding grammatical nuances and word sequencing. Diverging from the bag-of-words approach, TF-IDF assigns weights to words based on their relative significance, thereby attributing higher values to rare words compared to commonly occurring ones.

The conversion of textual data into numerical representations plays a critical role in the development of predictive tools through computational algorithms in the field of machine learning. Over the past decade, a multitude of scholarly efforts has been dedicated to the vectorisation or embedding of textual inputs, aiming to produce meaningful numerical representations. In the academic work by (Spichakova and Haav, 2020), the authors applied the technique of Doc2Vec, as introduced by Le and Mikolov in 2014 (Le and Mikolov, 2014), to represent commodity transactions. Unlike Word2Vec's focus on word-level embeddings,

Doc2Vec's main feature lies in its ability to encapsulate the semantic understanding of entire documents or transactions. Both methodologies share the common objective of converting textual data into numerical representations but generate embedding in a context-independent approach.

Convolutional Neural Networks (CNN) are another deep learning architecture that has been applied in diverse commodity classification tasks. For example, (He *et al.*, 2021) and (Zhou *et al.*, 2022) employ CNNs in the context of Chinese commodity declaration tasks. Their use of CNNs suggests the versatility of these architectures in handling textual data related to commodity descriptions, exemplifying CNNs' adaptability across diverse data modalities.

In parallel, the Long Short-Term Memory network (LSTM), a recurrent neural network (RNN) architecture proficient in capturing long-range dependencies in sequential data, featured prominently in the work of (Du *et al.*, 2021). They applied LSTM to model the sequential nature of commodity transactions, aiming to capture and encode the intricate sequential information within the transactional records. This use of LSTM underscores its suitability for tasks involving temporal data processing, such as transactional records analysis in commodity classification studies.

Transformer-based models have emerged as pivotal tools in academic research, showcasing substantial advancements over traditional RNN-based architectures. In particular, models like BERT (Devlin *et al.*, 2019) have demonstrated remarkable performance owing to their adeptness in capturing intricate syntactic and semantic representations, primarily facilitated by the integration of attention mechanisms (Vaswani *et al.*, 2017). In the domain-specific application for Portuguese goods classification, (de Lima *et al.*, 2022) effectively employed BERTimbau, a pre-trained BERT model tailored explicitly for Portuguese language tasks and a recent study conducted by (Lee *et al.*, 2023) employed KoBERT and KLUE, Korean pre-trained language models, as the backbone to propose predictions of HS codes derived from item descriptions and prior decisions, while also providing explanations for these predictions. This utilisation exemplifies the adaptability and efficacy of transformer-based models in addressing language-specific classification challenges.

Furthermore, the development of DistilBERT (Sanh *et al.*, 2020) marked a significant advancement, presenting a compelling alternative with comparable performance to BERT but with reduced computational demands due to its smaller model size. This innovation has contributed to enhancing efficiency, particularly in resource-constrained computational environments. The study conducted by (Anggoro *et al.*, 2023) showcased the implementation of the DistilBERT model within the domain of international trade transactions for commodity classification.

The application of sentence-level embeddings to product declarations has been previously explored through the implementation of Universal Sentence Encoder (USE) (Chen *et al.*, 2021) and SBERT (Pain, 2021). These studies employ an unsupervised learning approach to assess the accuracy of assigned HS codes by utilising similarity scores. In the study by (Chen *et al.*, 2021), USE is applied to convert historical product declarations into centroid vectors that represent each HS code. To evaluate this method, unlabelled product description vectors are compared against each HS code centroid vector in the lookup table, with the highest similarity score indicating the correct HS code. Conversely, the study by (Pain, 2021) involves matching a trade manifest against all descriptions in the commodity reference, with both sources vectorized using the SBERT model. The top k-matched commodities are then suggested as the correct code based on the cosine similarity score.

More recently, a study by (Navasardyan, 2024) employed a LLM, GPT-3, to determine the appropriate HS code by leveraging a prompting strategy. This approach offered the distinct advantage of providing explanations for the model's selection of the product category, enhancing interpretability in the classification process. However, the scalability of this method

for handling large volumes of transactions remains uncertain. Another study by (Lee *et al.*, 2024) proposes an explainable decision-support model that utilizes Korea transformer-based models as its backbone to facilitate HS code prediction. The predicted candidates are supplemented with confidence scores and highlighted product references to enhance interpretability.

The review presented above underscores that numerous model architectures have been systematically developed within the trade domain to facilitate efficient and accurate automated product classification. In light of the increasing focus on model performance in recent literature, this research seeks to contribute to these advancements by developing a model aimed at improving the accuracy of product classification in trade operations. Our proposed approach involves representing trade transactions at the sentence level and fine-tuning the model through a contrastive learning paradigm. This constrastive learning approach uses a novel loss function to generate transaction embeddings, thereby enhancing the overall performance of the model.

Based on the discussion above, several key distinctions of this study from previous works can be highlighted:

1. Firstly, this research focuses on extracting features to generate full sentence or transaction embeddings, unlike in the work of (Altaheri and Shaalan, 2020), which utilised TF-IDF. By adopting this approach, we aim to capture the semantic meaning in the product descriptions, taking into account the entire context rather than treating it as a collection of isolated words. This allows for more accurate classification, especially when products have multiple attributes that must be considered simultaneously.
2. In contrast to the approach of (Spichakova and Haav, 2020), which generates vector embeddings for entire documents using Doc2Vec or sliding convolutional filter pooled to form text representation (He *et al.*, 2021; Zhou *et al.*, 2022), our method employs LLMs transformer-based models, which have been trained with large corpora producing rich representations. Furthermore, our approach fine-tunes the transformer model at the sentence level using pairwise sentences and a contrastive learning method different from the work of (Anggoro *et al.*, 2023), which uses aggregated token to represent the text. This enables the embeddings to capture not only syntax but also the semantic meaning of product descriptions more effectively, making the transaction representations more comprehensive.
3. Additionally, our approach prioritises efficiency and flexibility by converting product descriptions into fixed-length embeddings, preferable for further downstream tasks. Using a supervised approach, the embeddings can be integrated with various classification algorithms, thereby creating more discriminative boundaries for classifying product descriptions. This approach is expected to yield higher performance compared to the unsupervised methods employed by (Chen *et al.*, 2021; Pain, 2021).

2.2. Sentence Representation

Conventional word embedding operates at the word level and often relies on padding to ensure uniform input sizes. The process involves averaging or pooling individual word embeddings to generate sentence-level embeddings from the input text. However, this method tends to obscure the semantic meaning encapsulated within the entire sentence. Conversely, in transformer-based models like BERT, the conventional methodology for generating complete sentence embeddings involves averaging the output of the final layer of the model or employing the [CLS] token. Nevertheless, this approach is criticised for its suboptimal performance in tasks that measure semantic similarity (Wang and Kuo, 2020).

SBERT (Reimers and Gurevych, 2019) provides a noteworthy example of sentence-level frameworks that leverage pre-trained transformer-based models as their backbone encoder. The architecture of SBERT comprises Siamese encoders crafted to encapsulate the

semantic essence of sentences via semantic textual similarity assessments. A pivotal facet of SBERT involves employing a pre-trained transformer model as the foundational structure for encoding textual inputs. The resulting output traverses an additional pooling layer, yielding a fixed-length dimensional vector. When paired with objective functions, this methodology involves feeding the network with pairwise or triple input sequences in order to update the shared weights of a bi-encoder network to capture the essence of sentence embeddings.

Numerous pre-trained SBERT models are readily accessible and can be directly employed for the generation of sentence embeddings. These pre-trained models have undergone rigorous training using meticulously labelled sentence pairs, such as those sourced from the Stanford Natural Language Inference dataset (Bowman *et al.*, 2015). However, an alternative avenue involves tailoring the SBERT framework's foundational elements during the training phase, such as configuring encoders and loss functions and employing domain-specific datasets, as undertaken in our study, rather than relying solely on existing pre-trained models. This methodology enables the customisation of pre-trained models to support the specific requirements of the domain, thereby enhancing the quality of embeddings utilised for subsequent classification tasks.

In the implementation of SBERT for HS code classification described in the study by (Pain, 2021), an existing pre-trained SBERT model is employed without additional fine-tuning to vectorize documents. These embeddings are then matched, based on semantic textual similarity, to a lookup table containing product references, which have been vectorized using the same model, to identify the top k-matched commodities associated with the HS code.

In contrast, our approach involves developing a custom SBERT model that is fine-tuned using domain-specific datasets configured in pairwise samples following a contrastive learning approach. This fine-tuning process is further refined through specific configuration settings during the training phase, including tailored loss functions, pooling layers, and hyperparameter optimization, with the aim of generating transaction embeddings that can be effectively utilised for downstream tasks, such as product classification, within a supervised learning framework.

2.3. Loss Function

The process of identifying and adjusting the loss function represents a pivotal stage in enhancing a model's performance (Wang *et al.*, 2022). Our investigation aims to assess the impact of combining a generated sentence embedding from a specific domain with a particular loss function on the subsequent task performance.

In the context of multi-class classification tasks, the combination of softmax and cross-entropy loss remains prevalent within deep learning architectures. The primary objective of using cross-entropy loss is to aid the model's label prediction ability. This loss function is favoured due to its robust theoretical underpinnings and widespread applicability, often standing as a strong competitor to alternative loss functions (Andreieva and Shvai, 2021). However, it is important to note certain limitations associated with cross-entropy loss, such as its susceptibility to noisy labels and poor margins, which consequently impact the model's generalisation performance (Khosla *et al.*, 2021; Pang *et al.*, 2020).

The ranking loss function, also recognised as contrastive loss or margin loss, operates with the main objective of learning an embedding space where pairs of similar samples maintain close proximity while dissimilar ones are positioned far apart. This aspect stands as a pivotal element in learning input representations. Contrastive learning is adaptable to both supervised and unsupervised data, with its prevalence evident in the self-supervised learning paradigm, specifically the augmentation of data within vision computing (Chen *et al.*, 2020). However, when applied to textual data, data augmentation poses inherent challenges due to the discrete nature of such data (Gao *et al.*, 2021). Effective augmentation of textual data

necessitates a comprehensive understanding of the data, considering that altering the data can significantly modify its semantic meaning.

Nevertheless, several methodologies have been employed to augment textual data within the domain of NLP. For instance, BART (Lewis *et al.*, 2020) implements a technique involving the random manipulation of word positions within sentences, while CERT (Fang *et al.*, 2020) facilitates back-translation as a means of augmenting input data. Recent advancements have explored augmentation techniques such as word deletion, reordering, and substitution specifically tailored for enhancing sentence representations (Wu *et al.*, 2020) and utilising text data augmentation approach based on ChatGPT (Dai *et al.*, 2023). Although, based on our knowledge, a systematic way of generating augmentation data is not currently available.

In addition to employing data augmentation techniques to optimise the efficacy of the contrastive loss function, the arrangement of input samples during training plays a critical role in optimising this objective function, particularly when utilising labels that facilitate the creation of pairwise or triplet samples. MNR loss (Henderson *et al.*, 2017) incorporates a learning strategy specifically designed to maximise the similarity between paired samples of input representations. This particular loss function necessitates inputs to be structured in pairs, with each pair comprising an anchor (the premise) and a corresponding positive sample. Hence, the presence of positive samples stands as a key component within this methodology, particularly in the context of supervised contrastive learning. This approach relies on labelled datasets to create training samples for the models under study (Moukafih *et al.*, 2022).

The fusion of SBERT with MNR loss presents a considerable enhancement of sentence representations as described by the SBERT author (Reimers, 2022). This contrast against the original SBERT method utilising Softmax loss emphasises the advancements achieved through this approach (Reimers and Gurevych, 2019). An additional adaptation within the scope of MNR loss is the Systematic MNR loss, which takes into account the mean of both forward and backward pairs, representing a further refinement within the spectrum of MNR loss variations.

3. Methodology

In this section, we describe our methodology for generating sentence embeddings for descriptions of products that appear in international trade transactions. These embeddings serve as the foundational elements for subsequent downstream tasks, particularly in the domain of product classification. Figure 1 shows the building blocks of our method.

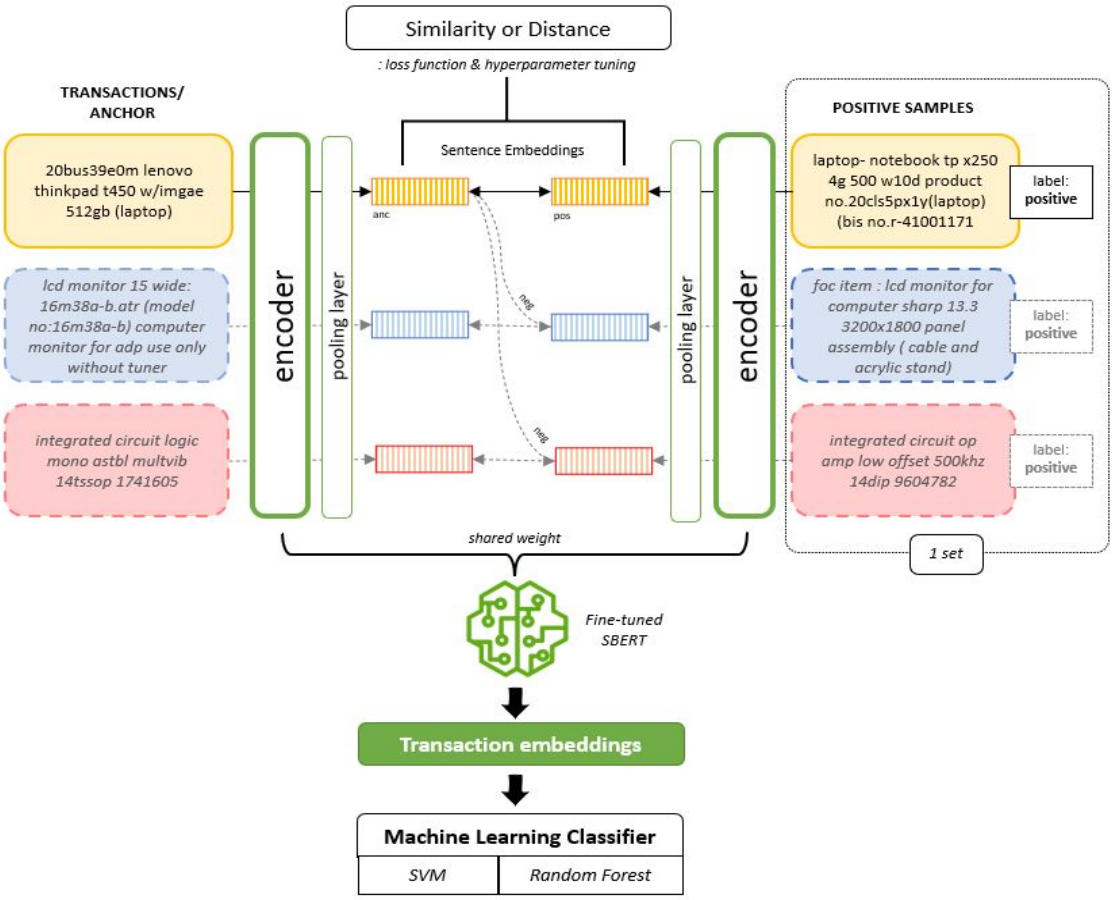

Figure 1. The proposed supervised contrastive learning process using the SBERT framework is illustrated. The transformer-model encodes product description, which is then passed through a pooling layer. During the training phase, anchor samples are paired with corresponding positive samples (from the same class). The model learns to minimise the distance between these anchor-positive pairs in the latent space while maximising the distance between the anchor and samples from other classes (negative samples). Finally, the fine-tuned model is employed to generate transaction embeddings, which are then utilised in a downstream classification task. Source: Author's own work.

3.1. Datasets

Numerous studies have addressed the task of HS code prediction, as evidenced by prior research endeavours (Altaheri and Shaalan, 2020; de Lima *et al.*, 2022). However, access to datasets relevant to this domain remains notably limited within the public domain. To address this limitation, we obtained two publicly available datasets from distinct sources: the 2016 India trade dataset (Zauba, 2023) and the 2018 U.S. trade dataset (Enigma, 2023). These datasets encompass thousands of products across a wide range of categories, providing a robust foundation for comprehensive exploration within the field. Additionally, since our backbone model, BERT and DistilBERT, have been trained on a large corpus of English text, utilising these English-language datasets offers significant advantages. This alignment facilitates the model's ability to generalise more effectively, thereby enhancing its potential to achieve optimal performance.

The datasets under consideration originate from customs declaration forms, as depicted in figure in Appendix I, encompassing a diverse array of elements. Among these attributes, particular focus is directed towards two key components: the product description and the HS code. The HS code assumes significance within this context as it serves as the ground truth or label for each transaction. This globally standardised coding system, as endorsed by the World Customs Organization, typically comprises a six-digit label widely utilised in trade

transactions. In particular, additional digits may be appended to this code, depending on local jurisdictional requirements. In our study, we specifically selected the initial two digits of HS codes, referred to as chapters (digits 84 and 85), encompassing products such as electrical machinery, televisions, personal computers, and mechanical appliances. These items are considered particularly challenging to classify (Lee *et al.*, 2023; Spichakova and Haav, 2020).

The distribution of classes within the chosen HS code chapters reveals a substantial imbalance, with an average ratio of 30:1 observed between the majority and minority classes across two distinct datasets. Specifically, these datasets pertain to telephone sets and bulldozers, constituting commodities for the major class, contrasting with battery chargers and agricultural machinery, representing the products of the minority class. A detailed breakdown of the dataset distribution is provided in Table I. For each dataset subset, we randomly allocated 70% of transactions for the fine-tuning of the SBERT model, reserving the remaining 30% for training and validation purposes within the machine learning classifier framework.

Datasets	Number of Classes	Number of Transactions
Datasets 1 (India commodity trade transactions)	172	66,522
Datasets 2 (U.S. commodity trade transactions)	112	58,003

Table I. The number of transactions and the corresponding commodity classes used in this study for the India and U.S. trade datasets.

The product descriptions within the transactions serve as textual narratives describing product details, covering not only the primary product identity but also its specifications, such as product type, serial number, weight, and dimensions, as seen in Table II. Additionally, these descriptions occasionally provide information about the product's condition or status. Moreover, a large number of product descriptions deviate from the conventions of natural language, lacking contextual coherence and frequently contain out-of-vocabulary terms.

Data pre-processing tasks frequently involve cleaning and manipulating textual inputs, a common practice in data preparation pipelines when dealing with textual data. Evidence from prior studies suggests that selecting an appropriate preprocessing task can enhance the performance of classification tasks (Uysal and Gunal, 2014). To investigate the efficacy of eliminating non-essential information in improving model performance within trade transactions, we conducted specialised preprocessing. This approach involved extracting and retaining crucial product-related details while discarding extraneous information. The preprocessing steps encompassed converting text into lowercase, removing duplicate transactions, cleaning alphanumeric characters, and preserving tokens with a minimum length of two characters. Gensim, an open-source software library, was employed to facilitate these preprocessing steps.

Table II illustrates examples covering both the original transactions and the outcomes subsequent to the cleaning procedures. According to this approach, there was an 8% reduction in the number of tokens within Dataset 1 and a 19% reduction within the second dataset. This step aimed to evaluate the impact of removing non-essential information on model performance, specifically customised for trade transactions.

HS Code	Commodity Description (Original)	Commodity Descrition (Pre-procecessed)
85423100	INTEGRATED CIRCUIT LOGIC MONO ASTBL MULTVIB 14TSSOP 1741605	INTEGRATED CIRCUIT LOGIC MONO ASTBL MULTVIB TSSOP
85423100	INTEGRATED CIRCUIT OP AMP LOW OFFSET 500KHZ 14DIP 9604782	INTEGRATED CIRCUIT OP AMP LOW OFFSET KHZ DIP
84713010	X2 LAPTOP LENOVO MODEL THINKPAD T430	LAPTOP LENOVO MODEL THINKPAD
84713010	LAPTOP- NOTEBOOK TP X250 4G 500 W10D PRODUCT NO.20CLS5PX1Y(LAPTOP) (BIS NO.R-41001171	LAPTOP NOTEBOOK TP PRODUCT NO CLS PX LAPTOP BIS NO
85285100	LED MONITOR, S22E360H, 21.5, INDIA, LB50/S22ECO, LS22E360HS/XL (22)(60 PCS) (FOR COMPUTER) (SAMSUNG)	LED MONITOR INDIA LB ECO LS HS XL PCS FOR COMPUTER SAMSUNG
84713010	20BUS39E0M LENOVO THINKPAD T450 W/IMGAE 512GB (LAPTOP)	BUS LENOVO THINKPAD IMGAE GB LAPTOP
85285100	LCD MONITOR 15 WIDE: 16M38A-B.ATR (MODEL NO:16M38A-B) COMPUTER MONITOR FOR ADP USE ONLY WITHOUT TUNER	LCD MONITOR WIDE ATR MODEL NO COMPUTER MONITOR FOR ADP USE ONLY WITHOUT TUNER
85285100	FOC ITEM : LCD MONITOR FOR COMPUTER SHARP 13.3 3200X1800 PANEL ASSEMBLY (CABLE AND ACRYLIC STAND)	FOC ITEM LCD MONITOR FOR COMPUTER SHARP PANEL ASSEMBLY CABLE AND ACRYLIC STAND

Table II. The comparison of the original product descriptions with their modified versions after the preprocessing stage. The modifications include removal of irrelevant information, which are commonly applied for the classification models.

3.2. SBERT and Pretrained Models

The enhancement of transformer-based models for generating sentence representations as our aim in this study is facilitated through the utilisation of the SBERT framework. The adopted fine-tuned model is expected to possess the capability to capture relationships between words, attributes, and concepts within product descriptions. This method of representation allows for the generation of encoded product descriptions that produce similar embeddings for transactions within the same category, thereby promoting logical grouping based on semantic similarities. Consequently, the clustered shared semantics of transactions have the potential to enhance discriminability, thereby improving the effectiveness of downstream tasks.

To instantiate our proposed framework, a collection of data comprising textual information along with corresponding ground truth labels (i.e., HS codes) for the textual data is imperative. Implementing the SBERT model requires the construction of pairwise samples derived from the international trade dataset, serving as the foundational element for model training. Our approach adopts a supervised methodology for constructing these pairwise samples, drawing inspiration from the work of (Moukafih *et al.*, 2022), where the experimental setup utilises labels derived from the data for binary classification tasks, particularly in sentiment analysis methods. However, our research deals with a more fine-grained and challenging classification problem involving numerous product categories spanning hundreds of classes.

To illustrate the construction of pairwise samples, it comprises two samples, referred to as an anchor sample and a positive sample. An anchor sample refers to a transaction extracted from the dataset, while the positive sample is a randomly selected transaction that shares the same class label as the anchor sample. Exploiting labels from datasets in constructing pairwise samples is chosen instead relying on conventional data augmentation methods which is often employed in contrastive learning frameworks. Our consideration of this approach is justified by the reason that modifying information within trade transactions negatively impacts the model's ability to produce accurate outcomes.

Figure 2.a. provides a visual representation of the methodology employed for constructing pairwise samples used for training the model. Positive samples, representing transactions sharing identical class labels as the anchor samples, are randomly selected from the training datasets. Initially, in our approach, each anchor within the training datasets is designated to possess solely one positive sample. However, as depicted in Figure 2.b, our

methodology is also flexible enough to allow an anchor to be associated with multiple positive samples.

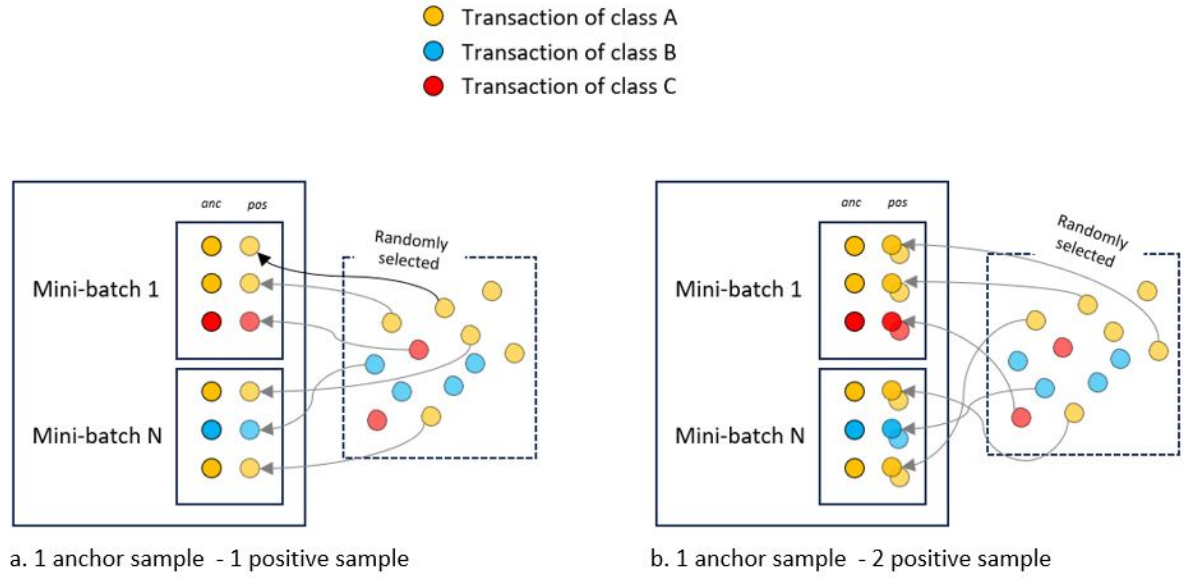

Figure 2. The figure demonstrates the pairwise sample construction process. For each anchor sample, a positive sample is randomly selected from the dataset, all belonging to the same class as the anchor (left). However, an anchor can have more than one positive samples, allowing the model to learn from diverse representations within the same class (right). Source: Author's own work.

In our experimental design, a designated backbone encoder, such as a pre-trained transformer-based model, is chosen as the core feature extractor to facilitate the embedding of textual data. Our decision not to utilise existing pre-trained SBERT models, as demonstrated in the work of (Pain, 2021), derives from our motivation to evaluate the performance of fine-tuned transformer-based models (Anggoro *et al.*, 2023) in comparison to our method of fine-tuning transformer-based models through a contrastive learning approach using SBERT framework. To achieve this, we employed BERT and DistilBERT as feature extractors for our datasets and then configured the dataset structure to facilitate the generation of pairwise samples. The systematic structuring of the dataset into paired instances laid the groundwork for training the SBERT model using customised data, with the primary objective of maximising its efficacy performance for the subsequent task.

The subsequent step involves the establishment of a standardised, consistent-sized sentence embedding. This is accomplished by integrating a pooling layer following obtaining the output from the transformers-based encoder, as illustrated in Figure 1. Diverse pooling strategies are available to facilitate this process. For instance, the MEAN strategy involves computing the mean of all vectors, while the MAX pooling strategy takes into account the component-wise maximum value among the output vectors. Another strategy uses the [CLS] token pooling method, which is derived from the final layer of the underlying pre-trained model. This token aims to encapsulate and aggregate word representations for the entire phrase. Additionally, an alternative option involves the concatenation of multiple pooling outputs, offering a flexible approach to yield customised fixed-length embeddings. These diverse pooling strategies enable the transformation of variable-length textual data into consistent, fixed-sized sentence embeddings, enhancing the utility and applicability of the generated representations within downstream tasks. Our intention is to apply a number of different pooling strategies to investigate their influence on the model's performance.

3.3. MNR Loss Method

Distance functions play a fundamental role in contrastive learning, serving to quantify embedded representations of pairwise samples. Commonly employed metrics encompass Manhattan distances, Euclidean distances, and similarity functions like cosine similarity and dot products. Furthermore, it is worth mentioning that while dot products can be influenced by vector size and direction, cosine similarity primarily hinges on the angle between vectorised representations. (Thakur *et al.*, 2021) observe that cosine similarity demonstrates a preference for shorter documents compared to the dot product, which tends to favour longer documents. In our experiment, we utilised both similarity metrics across various datasets originating from different countries.

When employing cosine similarity to compare paired samples, the resulting values tend to be relatively small. To overcome this, a scaling factor is introduced as a temperature parameter (Radford *et al.*, 2021). However, determining the optimal scale value remains an ongoing challenge. Some approaches involve setting the value at 1 and gradually increasing it over several training batches (Henderson *et al.*, 2020), while the SBERT's author advocates setting the scale value to 20, demonstrating efficacy, particularly in scenarios employing cross-entropy loss.

MNR loss is particularly suitable in scenarios where only positive samples are utilised as training datasets. This loss function necessitates inputs structured in a batch configuration, comprising n sentence pairs $[(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)]$, wherein randomly selected pairs (a_i, b_i) represent positive pairs, and n denotes the number of pairs. MNR loss aims to minimise the distance between similar pairs (a_i, b_i) while concurrently maximising the distance between dissimilar pairs (a_i, b_j), where $i \neq j$. This objective culminates in the eventual training outcome, attracting similar transactions closer together while concurrently pushing dissimilar transactions apart within the feature embedding space. Appendix II encapsulates a pseudocode representation of the MNR Loss.

3.3. Machine Learning Classifier

The refined representations acquired through the fine-tuned SBERT model capture semantic and syntactic features extracted from commodity transactions, projecting an optimistic influence on the subsequent performance of downstream tasks. Our next objective involves the application of traditional machine learning classifiers to undertake the classification task. Within our experimental paradigm, we employ two classifiers: SVM and Random Forest. Our consideration is to employ traditional machine learning classifiers, assuming that the models will continue to generate well-performing results due to the representative transaction embeddings generated through our methodology.

The classifiers are utilised in their standard configurations without further configuring their hyperparameters in our experiment. The selection of Random Forest and SVM is not only grounded in their recognition as superior traditional machine learning models (Fernández-Delgado *et al.*, 2014), but also is motivated by their proven efficacy in handling high-dimensional data, a characteristic prevalent in our dataset, and their extensive adoption within the machine learning domain for classification tasks. SVM, renowned for its ability to learn linear hyperplanes for data point classification, excels, particularly with high-dimensional data and multiple classes. SVM employs the kernel trick, such as Radial Basis Function (RBF) kernel, to handle nonlinear classification problems, thus demonstrating the substantial capability of achieving reliable classifications (Zareapoor *et al.*, 2018). On the other hand, Random Forest, an ensemble-based algorithm introduced by (Breiman, 2001), emerges as a classifier choice because of its proven competence in managing high-dimensional and noisy text data (Islam *et al.*, 2019). This algorithm leverages ensembles of trees to generate predictions, effectively reducing overfitting tendencies and mitigating biases inherent in such

datasets. In addition, the random forest algorithm is often regarded as a straightforward technique to tune in comparison to boosted algorithms, such as XGBoost and typically requires more training time, particularly during the prototyping phase.

4. Experiments

4.2. Experimental setup

The experimentation framework was conducted in Python3, employing diverse APIs, including PyTorch, HuggingFace, SBERT, scikit-learn, and additional tools such as the LIME package for interpretability. The computational infrastructure for conducting these experiments was provided by the cluster computing resources. The utilised environment featured GPU, complemented by high memory capacity, enabling efficient and robust computational processing for the experimental procedures.

4.3. Building Customised SBERT

This section presents the orchestration of building blocks within the SBERT framework, illustrating the configuration setup employed to fine-tune the network. Steps in this process include defining encoder and training sample configurations, establishing the loss function, and specifying various hyperparameters instrumental during the training phase.

4.3.1. Encoders, Pooling Strategy and Positive Samples

Pre-trained transformer-based models, specifically BERT and DistilBERT, are sourced from the HuggingFace hub to serve as encoders tasked with vectorising textual inputs. Both variants of these pre-trained models - being the base models and uncased or case-insensitive - were employed in our study. Regarding the pooling strategy, we evaluate each individual strategy, including MEAN, MAX, and the specialised [CLS] token pooling strategy. This configuration resulted in the generation of 768-dimensional features for each product transaction embedding.

A dedicated function is developed to facilitate the creation of pairwise sentences, where each pair comprises an anchor and a positive sample aimed at training the SBERT model. Within this function, positive samples were randomly selected without replacement, thereby organising the training datasets to feed the Siamese network. This methodology produces 46,565 pairs for Dataset 1 and 39,612 pairs for Dataset 2. Initially configured with one positive sample for one anchor, our experimentation also involves exploring the impact of assigning three positive samples for one anchor. This adjustment expanded the number of pairs, allowing an investigation into the correlation between model performance and increased sample number despite the longer training duration incurred with a larger number of samples.

Following the completion of training samples, model training is conducted in batches for multiple pairs (anchor, positive) in parallel. Batch sizes of 8 and 32 are evaluated to identify potential enhancement in model performance with larger batch sizes. This approach draws inspiration from the established effectiveness of increased batch sizes in contrastive learning.

4.3.2. MNR Loss Experiment

Our aim involves the establishment of diverse parameter configurations associated with the loss function. These configurations include adjusting the scale parameter, the selection of a suitable similarity function, and the application of the Symmetric MNR loss function, which computes the average of forward and backward loss values. The chosen similarity functions for measuring distances between sentence embeddings include cosine similarity and dot product. In the absence of a comprehensive systematic study offering insights into an optimal scaling factor to amplify the magnitude of cosine-similarity scores, we initiate our

investigations with a scaling factor of 20. The square root of the dimensional feature size, which is 23 in this study (Henderson *et al.*, 2020), is utilised as a reference metric. Subsequently, we incrementally increase the scale parameter to 50 to systematically explore its influence on the model's performance and identify changes in model behaviour and efficacy.

4.3.3. Hyperparameters

Additionally, a range of hyperparameter adjustments is systematically introduced during the training phase to enhance the algorithm's efficacy, a strategy commonly employed for optimisation purposes (Weerts *et al.*, 2020). The batch size, a critical parameter, is varied between 8 and 32, meticulously selected to match the computational memory capacity. The number of epochs for the training phase is explored across 10 and 30 iterations, impacting the depth and extent of model training. Finally, the learning rate is examined within the range of $2e^{-5}$ and $5e^{-5}$ using AdamW as optimizer, responsible for the magnitude of adjustments made to model weights during training.

5. Results and Discussion

This section outlines the results of the experiments conducted across various settings. In this analysis, we present the model performance of two machine learning algorithms - SVM and Random Forest classifiers. These algorithms leverage features generated from the fine-tuned SBERT model.

5.1. Model Training and Metrics

In our evaluation process, we adopt the k-fold cross-validation methodology to assess the classifier's performance. The datasets were partitioned into 5 folds, using a stratified cross-validation approach, ensuring each fold preserved a balanced representation across classes. This strategy is effective in handling the inherent imbalances within the datasets, thereby fostering robust evaluation outcomes.

To comprehensively evaluate model performance, we use Cohen's Kappa, macro-average, micro-average, and weighted average for precision, recall, and F1-score. Cohen's Kappa is a metric used for evaluating the performance of classification models that can be applied to text data (Kolesnyk and Khairova, 2022). Macro-average treats each class equally, while micro-average evaluates performance globally by aggregating all instances. The weighted average adjusts for class imbalance by factoring in class sizes. These metrics ensure a balanced and thorough assessment of our model's effectiveness across different class distributions. Particularly, our emphasis resides on the F1 score due to its robustness in handling imbalanced datasets, providing a balanced assessment of the classifier's performance in scenarios with unequal class distributions.

5.2. Performance of Model and Model Size

Table III illustrates a marked enhancement in model performance resulting from our proposed method as compared to the fine-tuning transformer-based model conducted in this study (Anggoro *et al.*, 2023). Specifically, our model exhibits substantial improvements in the weighted average F1 score, elevating it from 0.8406 to 0.8620 when employing SVM and achieving 0.8656 with the Random Forest algorithm for Dataset 1. Similar notable improvements are observed in Dataset 2, demonstrating a 4% increase in weighted average F1 score with both SVM and Random Forest classifiers compared to the most optimal results attained through fine-tuning the transformer models—BERT or DistilBERT.

Regarding the model size implications, employing the SBERT with DistilBERT as encoder in tandem with SVM, produces a model size comparable to that of the fine-tuned

DistilBERT model. However, the utilisation of SBERT + BERT as encoder in conjunction with the Random Forest algorithm results in an expanded model size exceeding that of fine-tuning the BERT model. These findings underscore the intricate trade-offs between model performance enhancements and resultant model sizes associated with distinct algorithmic configurations.

Dataset	Method	Cohen Kappa	Macro-average			Micro-average			Weighted average			Model Size
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	
1	Fine-tuning BERT	0.8406	0.8444	0.8426	0.8462	0.8097	0.8426	0.8426	0.8181	0.8426	**0.8406**	418 MB
	Fine-tuning DistilBERT	0.8386	0.8383	0.8407	0.8428	0.8088	0.8407	0.8407	0.8183	0.8407	0.8390	255 MB
	$SBERT_{enc.BERT}$ + SVM	0.8611	0.8700	0.8629	0.8686	0.8370	0.8629	0.8629	0.8464	0.8629	**0.8620**	449 MB
	$SBERT_{enc.DistilBERT}$ + SVM	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603	291 MB
	$SBERT_{enc.BERT}$ + RF	0.8647	0.8691	0.8664	0.8711	0.8445	0.8664	0.8664	0.8512	0.8664	**0.8656**	631 MB
	$SBERT_{enc.DistilBERT}$ + RF	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637	511 MB
2	Fine-tuning BERT	0.7680	0.7508	0.7711	0.7750	0.7127	0.7711	0.7711	0.7210	0.7711	0.7689	418 MB
	Fine-tuning DistilBERT	0.7626	0.7398	0.7658	0.7677	0.7032	0.7658	0.7658	0.7130	0.7658	0.7630	255 MB
	$SBERT_{enc.BERT}$ + SVM	0.7982	0.7881	0.8009	0.8090	0.7603	0.8009	0.8009	0.7681	0.8009	0.8013	437 MB
	$SBERT_{enc.DistilBERT}$ + SVM	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953	278 MB
	$SBERT_{enc.BERT}$ + RF	0.7972	0.7846	0.7998	0.8077	0.7624	0.7998	0.7998	0.7681	0.7998	0.8005	512 MB
	$SBERT_{enc.DistilBERT}$ + RF	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958	371 MB

Table III. The metric highlights the improved performance between baseline models (Fine-tuned BERT and DistilBERT) and fine-tuned SBERT combined with SVM and Random Forest classifiers. Key metrics such as Cohen's Kappa, along with macro, micro, and weighted averages of precision, recall, and F1 score, demonstrate that the fine-tuned SBERT-based models combining with SVM and Random Forest outperform the baselines in terms of classification performance.

5.3. Analysis of Model Explanation

Table IV demonstrates a substantial decline in model performance for F1 score ranging from 6% to 8% for both datasets' consequent to the elimination of certain information during the pre-processing phase. This reduction in performance suggests that the conventional pre-processing pipeline, often incorporated with the removal of alphanumeric details, may not be directly transferable or suitable for trade transactions within this context.

Dataset	Method	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	Pre-processed data	0.7803	0.804	0.7831	0.7923	0.7402	0.7831	0.7831	0.757	0.7831	**0.7790**
	Original data	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	**0.8603**
	Random Forest										
	Pre-processed data	0.8007	0.8225	0.8033	0.8121	0.7690	0.8033	0.8033	0.7848	0.8033	**0.8017**
	Original data	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	**0.8637**
2	***SVM***										
	Pre-processed data	0.7143	0.7196	0.7183	0.7338	0.6621	0.7183	0.7183	0.6795	0.7183	**0.7185**
	Original data	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	**0.7953**
	Random Forest										
	Pre-processed data	0.7165	0.7119	0.7203	0.7318	0.665	0.7203	0.7203	0.6788	0.7203	**0.7203**
	Original data	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	**0.7958**

Table IV. The model performance across different data pre-processing techniques. Key metrics such as weighted average precision, recall, and F1-score demonstrate how different pre-processing methods impact classification accuracy.

In order to offer a more comprehensive explanation for the model's behaviour in HS code prediction, we employed the LIME explanation method (Ribeiro *et al.*, 2016) to conduct an illustrative case study. We opted for LIME over alternative model interpretability techniques, such as SHAP, due to its stability and better for human interpretation, aligning with the context of our study, particularly in its application to Random Forest (Man and Chan, 2020). Figure 3 illustrates the comparison between the model trained to utilise the original product descriptions and its counterpart trained on modified datasets resulting from specialised preprocessing treatments. In particular, the model exhibits a higher prediction probability in determining the category when utilising the original text in contrast to the modified text. This discrepancy in predictive performance underscores the significance of specific attributes within the commodity description, such as the explicit mention of "*512 GB*", which impacts the model's decision-making process regarding label assignment to the description.

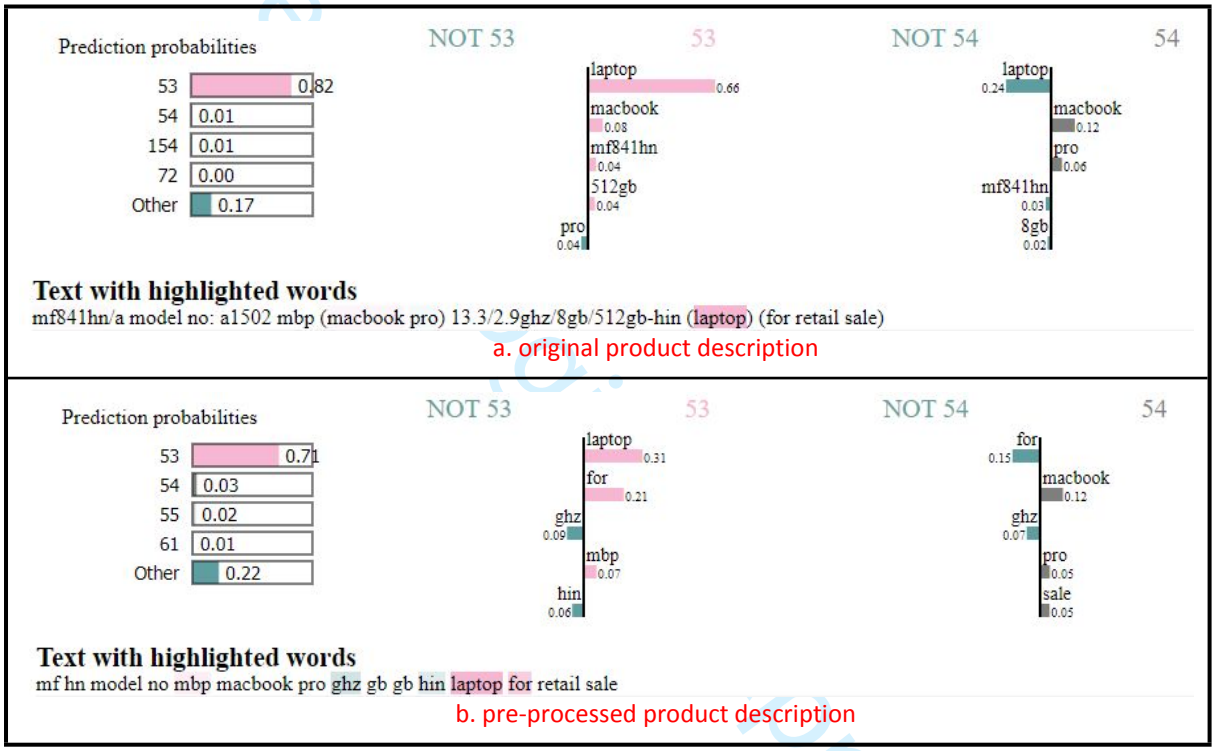

Figure 3. This figure compares the LIME visualisations of model predictions using different data pre-processing techniques. The visualisations indicate higher prediction probabilities for the original text (a), suggesting that the model's confidence in its predictions decreases after certain preprocessing steps (b). This observation implies that retaining specific textual information during preprocessing may enhance prediction accuracy.

As shown in Table V, for the results of using one and three positive samples, it becomes apparent that augmenting the number of positive samples paired with an anchor influences the model's performance. Broadly observed across both datasets, an increase in the number of positive samples demonstrates an improvement in the weighted average F1 score within the range of 2% to 3%. These outcomes, consistent across the SVM and Random Forest algorithms, indicate the advantages of increased positive sample pairing on the model's predictive efficacy for both datasets. Nevertheless, it is assumed that there exists an optimal quantity of positive pairs necessary to attain peak performance, after which further increases may not yield additional improvements.

Dataset	Samples	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	1 anchor, 1 positive	0.8365	0.8485	0.8386	0.8458	0.7965	0.8386	0.8386	0.8105	0.8386	0.8360
	1 anchor, 3 positive	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Random Forest										
	1 anchor, 1 positive	0.8412	0.8552	0.8432	0.8492	0.8051	0.8432	0.8432	0.8200	0.8432	0.8410
	1 anchor, 3 positive	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
2	***SVM***										
	1 anchor, 1 positive	0.7706	0.7621	0.7738	0.7847	0.7165	0.7738	0.7738	0.7304	0.7738	0.7735
	1 anchor, 3 positive	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Random Forest										
	1 anchor, 1 positive	0.7668	0.7517	0.7699	0.7770	0.7137	0.7699	0.7699	0.7244	0.7699	0.7688
	1 anchor, 3 positive	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958

Table V. The table compares the impact of using different numbers of positive samples per anchor on model performance. Some metrics are used to demonstrate how increasing the number of positive samples improves classification accuracy across two datasets.

The experimentation also provides insights into the outcomes derived from varying configurations of the loss function, such as strategies like MEAN, MAX, and [CLS] token. On the whole, Table VI shows the result of the implementation of the pooling strategies based on the experimental findings. The absence of a dominating pooling strategy across the experimental outcomes underscores the unavailability of a clear distinction among the pooling strategies concerning their impact on model performance.

Dataset	Pooling Strategy	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	CLS	0.8594	0.8696	0.8612	0.8675	0.8300	0.8612	0.8612	0.8425	0.8612	0.8603
	Mean	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Max	0.8564	0.8637	0.8582	0.8645	0.8269	0.8582	0.8582	0.8377	0.8582	0.8574
	Random Forest										
	CLS	0.8611	0.8676	0.8629	0.8686	0.8359	0.8629	0.8629	0.8453	0.8629	0.8621
	Mean	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Max	0.8593	0.8618	0.8611	0.8672	0.835	0.8611	0.8611	0.8414	0.8611	0.8605
2	***SVM***										
	CLS	0.7963	0.7991	0.7990	0.8114	0.7555	0.7990	0.7990	0.7690	0.7990	0.8006
	Mean	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Max	0.7927	0.7866	0.7955	0.8068	0.7529	0.7955	0.7955	0.7629	0.7955	0.7966
	Random Forest										
	CLS	0.7957	0.7857	0.7984	0.8084	0.7561	0.7984	0.7984	0.7654	0.7984	0.7997
	Mean	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Max	0.7909	0.7812	0.7937	0.8038	0.7513	0.7937	0.7937	0.7594	0.7937	0.7945

Table VI. The table compares model performance using various data pooling strategies, including CLS token, MEAN pooling, and MAX pooling. The results show that all three strategies yield similar or comparable performance across key metrics.

Table VII shows the implications arising from various configurations of the loss function on the model's performance. In the initial comparison, employing the Symmetric MNR loss showcases similar performance in the weighted average F1 score when compared

with the MNR loss function across both datasets. Furthermore, augmenting the scale parameter for the similarity function does not substantially affect the model's performance. Additionally, the adoption of a distinct similarity function, specifically the dot-product, as an alternative to the cosine similarity demonstrates decreased performance for Dataset 1 and relatively comparable results for Dataset 2.

Dataset	Loss Function Parameters	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Symmetric MNR Loss	0.8609	0.8722	0.8627	0.8686	0.8342	0.8627	0.8627	0.8455	0.8627	0.8616
	Scale - 50	0.8579	0.8679	0.8597	0.8661	0.8268	0.8597	0.8597	0.8388	0.8597	0.8585
	Dot Product	0.8512	0.8624	0.8531	0.8585	0.8135	0.8531	0.8531	0.8278	0.8531	0.8508
	Random Forest										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Symmetric MNR Loss	0.8638	0.8702	0.8655	0.8707	0.8389	0.8655	0.8655	0.8478	0.8655	0.8645
	Scale - 50	0.8612	0.8687	0.8630	0.8688	0.8340	0.8630	0.8630	0.8440	0.8630	0.8620
	Dot Product	0.8569	0.8675	0.8587	0.8641	0.8279	0.8587	0.8587	0.8397	0.8587	0.8574
2	***SVM***										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Symmetric MNR Loss	0.7968	0.787	0.7996	0.809	0.7531	0.7996	0.7996	0.7638	0.7996	0.8002
	Scale - 50	0.7905	0.7906	0.7933	0.8056	0.7469	0.7933	0.7933	0.7606	0.7933	0.7947
	Dot Product	0.7911	0.7863	0.7939	0.8084	0.7392	0.7939	0.7939	0.7533	0.7939	0.7954
	Random Forest										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Symmetric MNR Loss	0.7957	0.7782	0.7984	0.8068	0.7546	0.7984	0.7984	0.7606	0.7984	0.7989
	Scale - 50	0.7906	0.7856	0.7934	0.804	0.7518	0.7934	0.7934	0.7618	0.7934	0.7944
	Dot Product	0.7945	0.7875	0.7973	0.8057	0.7519	0.7973	0.7973	0.7617	0.7973	0.7974

Table VII. The table presents the performance comparison of the model with various configurations of the loss function. Key metrics are evaluated, showing how different loss functions settings impact model accuracy.

5.4. Parameters Importance

Multiple adjustments in hyperparameters, encompassing variations in batch size, the number of epochs, and the learning rate, are implemented during the fine-tuning of SBERT to observe their impacts on downstream tasks.

As depicted in Table VIII, the analysis of model training parameters reveals different impacts on optimising the weighted average F1 score. Specifically, employing a larger batch size appears to yield a modestly positive effect on enhancing the weighted average F1 score. Furthermore, augmenting the training iterations or epochs results in approximately 2% improvement in the weighted average F1 scores for SVM and Random Forest classifiers. In particular, varying the learning rate exhibits disparate outcomes: a lower learning rate ($2e^{-5}$) demonstrates better outcomes for Dataset 1 and Dataset 2, as observed across both SVM and Random Forest classifiers.

Dataset	Hyperparameters Training	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	Batch 8	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Batch 32	0.8619	0.8805	0.8637	0.8699	0.8359	0.8637	0.8637	0.8504	0.8637	0.8626
	Epoch 10	0.8411	0.8523	0.8432	0.8499	0.7980	0.8432	0.8432	0.8129	0.8432	0.8402
	Epoch 30	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Learning Rate 2e-5	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Learning Rate 5e-5	0.8488	0.8554	0.8507	0.8557	0.8228	0.8507	0.8507	0.8326	0.8507	0.8496
	Random Forest										
	Batch 8	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Batch 32	0.8667	0.8834	0.8684	0.8736	0.8416	0.8684	0.8684	0.8554	0.8684	0.8673
	Epoch 10	0.8475	0.8525	0.8494	0.8547	0.8136	0.8494	0.8494	0.8242	0.8494	0.8476
	Epoch 30	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Learning Rate 2e-5	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Learning Rate 5e-5	0.8471	0.8515	0.8491	0.8541	0.8213	0.8491	0.8491	0.8299	0.8491	0.8481
2	***SVM***										
	Batch 8	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Batch 32	0.7937	0.8017	0.7964	0.8093	0.7513	0.7964	0.7964	0.7670	0.7964	0.7977
	Epoch 10	0.7779	0.7669	0.7809	0.7916	0.7189	0.7809	0.7809	0.7320	0.7809	0.7795
	Epoch 30	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Learning Rate 2e-5	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Learning Rate 5e-5	0.7886	0.7824	0.7914	0.8026	0.7510	0.7914	0.7914	0.7603	0.7914	0.7928
	Random Forest										
	Batch 8	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Batch 32	0.7939	0.7864	0.7967	0.8062	0.7506	0.7967	0.7967	0.7619	0.7967	0.7972
	Epoch 10	0.7813	0.7658	0.7842	0.7922	0.7321	0.7842	0.7842	0.7418	0.7842	0.7839
	Epoch 30	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Learning Rate 2e-5	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Learning Rate 5e-5	0.7854	0.7716	0.7882	0.7974	0.7473	0.7882	0.7882	0.7536	0.7882	0.7891

Table VIII. The table presents the impact of various hyperparameter configurations on model performance. Key metrics are employed to assess the effectiveness of different learning rates, batch sizes, and epochs. The results show that optimal hyperparameter tuning can significantly improve the model's classification accuracy and overall performance.

5. Conclusions

This paper presents a method leveraging the SBERT framework for transaction embedding generation using international trade transaction datasets. The approach involves training a Siamese model network through pairwise samples and MNR loss, enhancing transaction representations. These resultant embeddings are integrated into traditional machine learning models like SVM and Random Forest, exhibiting enhanced performance over fine-tuned transformer-based models in handling international trade transactions. In addition, the experimentation provides valuable insights into transaction pre-processing and hyperparameter selection. While generating embeddings through this method might entail substantial computational resources, its efficacy in enhancing downstream task performance remains significant.

Moreover, future investigation is essential to explore practical applications, particularly in domains such as anomaly detection on international trade transactions. The inherent nature of the feature embeddings generated by this method tends to cluster transactions with analogous attributes within a shared embedding space. Utilising the clustering property of embeddings

holds promise for identifying anomalous transactions through distance analysis, signifying a potential opportunity for practical implementation in anomaly detection systems.

References

Altaheri, F. and Shaalan, K. (2020), "Exploring Machine Learning Models to Predict Harmonized System Code", in Themistocleous, M. and Papadaki, M. (Eds.), *Information Systems*, Springer International Publishing, Cham, pp. 291–303, doi: 10.1007/978-3-030-44322-1_22.

Andreieva, V. and Shvai, N. (2021), "Generalization of Cross-Entropy Loss Function for Image Classification", *Mohyla Mathematical Journal*, Vol.3, pp.3-10, doi: 10.18523/2617-7080320203-10.

Anggoro, A., Corcoran, P., Dennis, D.W. and Li, Y. (2023), "Using DistilBERT to Assign HS Codes to International Trading Transactions", *Information Systems and Technologies WorldCIST 2023 Volume 3*, Vol. Volume 3, presented at the World Conference on Information Systems and Technologies, Springer Cham, Pisa, Italy. available at: https://orca.cardiff.ac.uk/id/eprint/155999/ (accessed 1 August 2023).

Bowman, S.R., Angeli, G., Potts, C. and Manning, C.D. (2015), "A Large Annotated Corpus for Learning Natural Language Inference", in Màrquez, L., Callison-Burch, C. and Su, J. (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, doi: 10.18653/v1/D15-1075.

Breiman, L.E.O. (2001), "Random Forests", *Machine Learning*, Vol. 45, pp. 5–32, doi: 10.1023/A:1010933404324.

Chen, H., van Rijnsoever, B., Molenhuis, M., van Dijk, D., Tan, Y.H. and Rukanova, B. (2021), "The use of machine learning to identify the correctness of HS Code for the customs import declarations", *2021 IEEE 8th International Conference on Data Science and Advanced Analytics*, pp. 1–8, doi: 10.1109/DSAA53316.2021.9564203.

Chen, T., Kornblith, S., Norouzi, M. and Hinton, G. (2020), "A Simple Framework for Contrastive Learning of Visual Representations", *Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607.

Dai, H., Liu, Z., Liao, W., Huang, X., Cao, Y., Wu, Z., Zhao, L., *et al.* (2023), "AugGPT: Leveraging ChatGPT for Text Data Augmentation", doi: 10.48550/ARXIV.2302.13007.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019), "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *2019 Conference of The North American Chapter of The Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

Du, S., Wu, Z., Wan, H. and Lin, Y. (2021), "HScodeNet: Combining Hierarchical Sequential and Global Spatial Information of Text for Commodity HS Code Classification", *Advances in Knowledge Discovery and Data Mining,* pp. 676–689, doi: 10.1007/978-3-030-75765-6_54.

Enigma (2018), US Imports - Automated Manifest System (AMS) Shipments 2018, at: https://aws.amazon.com/ (accessed 1 August 2023).

Fang, H., Wang, S., Zhou, M., Ding, J. and Xie, P. (2020), "CERT: Contrastive Self-supervised Learning for Language Understanding", doi: 10.48550/arXiv.2005.12766.

Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014), "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?", *Journal of Machine Learning Research*, Vol. 15 No. 90, pp. 3133–3181.

Gao, T., Yao, X. and Chen, D. (2021), "SimCSE: Simple Contrastive Learning of Sentence Embeddings", *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, doi: 10.18653/v1/2021.emnlp-main.552.

He, M., Wang, X., Zou, C., Dai, B. and Jin, L. (2021), "A Commodity Classification Framework Based on Machine Learning for Analysis of Trade Declaration", *Symmetry*, Vol. 13 No. 6, pp. 964, doi: 10.3390/sym13060964.

Henderson, M., Al-Rfou, R., Strope, B., Sung, Y., Lukacs, L., Guo, R., Kumar, S., *et al.* (2017), "Efficient Natural Language Response Suggestion for Smart Reply, doi: 10.48550/arXiv.1705.00652".

Henderson, M., Casanueva, I., Mrkšić, N., Su, P.-H., Wen, T.-H. and Vulić, I. (2020), "ConveRT: Efficient and Accurate Conversational Representations from Transformers", in Cohn, T., He, Y. and Liu, Y. (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2161–2174, doi: 10.18653/v1/2020.findings-emnlp.196.

Islam, M.Z., Liu, J., Li, J., Liu, L. and Kang, W. (2019), "A Semantics aware random forest for text classification", *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1061–1070, doi: 10.1145/3357384.3357891.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., *et al.* (2021), "Supervised Contrastive Learning", arXiv, 10 March, doi: 10.48550/arXiv.2004.11362.

Kolesnyk, A.S. and Khairova, N.F. (2022), "Justification for the Use of Cohen's Kappa Statistic in Experimental Studies of NLP and Text Mining", *Cybernetics and Systems Analysis*, Vol. 58 No. 2, pp. 280–288, doi: 10.1007/s10559-022-00460-3.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D. (2019), "Text classification algorithms: A survey", *Information-an International Interdisciplinary Journal*, Vol. 10 No. 4, doi: 10.3390/info10040150.

Le, Q. and Mikolov, T. (2014), "Distributed Representations of Sentences and Documents", *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, pp.1188-1196, doi: 10.48550/arXiv.1405.4053.

Lee, E., Kim, S., Kim, S., Jung, S., Kim, H. and Cha, M. (2024), "Explainable Product Classification for Customs", *ACM Transactions on Intelligent Systems and Technology*, Vol 15 No.2, pp.25:1–25:24, doi: 10.1145/3635158.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., *et al.* (2020), "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension", *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, doi: 10.18653/v1/2020.acl-main.703.

de Lima, R.R., Fernandes, A.M.R., Bombasar, J.R., da Silva, B.A., Crocker, P. and Leithardt, V.R.Q. (2022), "An Empirical Comparison of Portuguese and Multilingual BERT Models for Auto-Classification of NCM Codes in International Trade", *Big Data and Cognitive Computing*, Vol. 6 No. 1, doi: 10.3390/bdcc6010008.

Man, X. and Chan, E. (2020), "The best way to select features?", *The Journal of Financial Data Science Winter 2021*, Vol. 3 No. 1, pp. 127–139, doi: 10.3905/jfds.2020.1.047.

Moukafih, Y., Ghanem, A., Abidi, K., Sbihi, N., Ghogho, M. and Smaili, K. (2022), "SimSCL: A Simple Fully-Supervised Contrastive Learning Framework for Text Representation", in Long, G., Yu, X. and Wang, S. (Eds.), *AI 2021: Advances in Artificial Intelligence*, pp. 728–738, doi: 10.1007/978-3-030-97546-3_59.

Navasardyan, Z. (2024), "Interpretable and Generalizable HTS Code Classification Framework", *Economics, Finance and Accounting*, Vol. 1 No. 13, pp. 140–140, doi: 10.59503/29538009-2024.1.13-140.

Pain, K. (2021), Harmonized System Code Classification Using Transfer Learning with Pre-Trained Weights, Dalhousie University, Halifax, 1 September.

Pang, T., Xu, K., Dong, Y., Du, C., Chen, N. and Zhu, J. (2020), "Rethinking Softmax Cross-Entropy Loss for Adversarial Robustness", *8th International Conference on Learning Representations, ICLR 2020*, Vol. 10, pp. 1–19, doi: 10.48550/arXiv.1905.10626.

Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., *et al.* (2021), "Learning Transferable Visual Models from Natural Language Supervision", in Meila, M. and Zhang, T. (Eds.), *International Conference on Machine Learning*, pp. 8748-8763.

Reimers, N. (2022), "Sentence Transformers NLI Training Readme, GitHub", *GitHub*, 19 December, available at: https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/nli (accessed 1 January 2024).

Reimers, N. and Gurevych, I. (2019), "Sentence-BERT: Sentence embeddings using siamese BERT-networks", *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 3982–3992, doi: 10.18653/v1/d19-1410.

Ribeiro, M.T., Singh, S. and Guestrin, C. (2016), "'Why Should I Trust You?': Explaining the Predictions of Any Classifier", *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, doi: 10.48550/arXiv.1602.04938.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2020), "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", paper accepted at *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*, doi: 10.48550/arXiv.1910.01108.

Singh, A.K. and Sahu, R. (2004), "Decision Support System for HS Classification of Commodities", *Proceedings of the 2004 IFIP International Conference on Decision Support Systems*.

Spichakova, M. and Haav, H.-M. (2020), "Using Machine Learning for Automated Assessment of Misclassification of Goods for Fraud Detection", in Robal, T., Haav, H.-M., Penjam, J. and Matulevičius, R. (Eds.), *Databases and Information Systems*, pp. 144–158, doi: 10.1007/978-3-030-57672-1_12.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A. and Gurevych, I. (2021), "BEIR: A Heterogeneous Benchmark for Zero-Shot Evaluation of Information Retrieval Models", *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track,* doi: doi.org/10.48550/arXiv.2104.08663.

Uysal, A.K. and Gunal, S. (2014), "The Impact of Preprocessing on Text Classification", *Information Processing & Management*, Vol. 50 No. 1, pp. 104–112, doi: 10.1016/j.ipm.2013.08.006.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., *et al.* (2017), "Attention Is All You Need", *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp.6000-6010, doi: 10.48550/arXiv.1706.03762.

Wang, B. and Kuo, C.-C.J. (2020), "SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 28, pp. 2146–2157, doi: 10.1109/TASLP.2020.3008390.

Wang, Q., Ma, Y., Zhao, K. and Tian, Y. (2022), "A Comprehensive Survey of Loss Functions in Machine Learning", *Annals of Data Science*, Vol. 9 No. 2, pp. 187–212, doi: 10.1007/s40745-020-00253-5.

Weerts, H.J.P., Mueller, A.C. and Vanschoren, J. (2020), "Importance of Tuning Hyperparameters of Machine Learning Algorithms", doi:10.48550/arXiv.2007.07588.

World Customs Organization. (2013), "HS classification handbook", at: http://harmonizedsystem.wcoomdpublications.org/pdfs/WCOOMD_MSH_EN.pdf (accessed 1 January 2023).

World Customs Organization. (2022), *WCO Annual Report 2022-2023*, World Customs Organization, at: https://www.wcoomd.org/-/media/wco/public/global/pdf/about-us/annual-reports/annual-report-2022_2023.pdf (accessed 1 August 2023).

Wu, Z., Wang, S., Gu, J., Khabsa, M., Sun, F. and Ma, H. (2020), "CLEAR: Contrastive Learning for Sentence Representation", *ACM Transactions on Intelligent Systems and Technology*, Volume 14, pp.1-34, doi: 10.48550/arXiv.2012.15466.

Zareapoor, M., Shamsolmoali, P., Kumar Jain, D., Wang, H. and Yang, J. (2018), "Kernelized Support Vector Machine with Deep Learning: An Efficient Approach for Extreme Multiclass Dataset", *Pattern Recognition Letters*, Vol. 115, pp. 4–13, doi: 10.1016/j.patrec.2017.09.018.

Zauba (2016), India HS Code Data, at: https://www.zauba.com/ (accessed 1 August 2023).

Zhou, C., Che, C., Zhang, X.S., Zhang, Q. and Zhou, D. (2022), "Harmonized system code prediction of import and export commodities based on Hybrid Convolutional Neural Network with Auxiliary Network", *Knowledge-Based Systems*, Vol. 256, p. 109836, doi: 10.1016/j.knosys.2022.109836.

Appendix I: An Example of Declaration Form

The figure below presents an example of declaration form, with key fields such as the product description and HS codes highlighted. These fields are crucial for the dataset used in this study, where the product description serves as the input for classification tasks, and the HS codes are the target labels for evaluating model performance.

Detailed description of contents	Custom duty (€)	Tax (€)	Qty	Net weight (kg)	Unit Value (€)	Total value (€)	HS tariff Nr.	Country of origin
Lafuma backpack for men hiking or skiing Volume: 40 L - Dim L: 100 cm - 50 cm	0.00	0.00	1	1.00	100	100	420292	FR
			Total Value	1.00	100	100	Shipping Value: 29.42 (€)	

Service required

☐ Gift ☒ Commercial Sample ☐ Commercial Sample

☐ Document ☐ Return ☐ Other (explain)

Insurance value : 0.00 €

Office of origin / date of posting

GENNEVILLIERS PFC - 10/06/2020

Appendix II: Pseudocode of Implementation MNR Loss

Algorithm 1 presents the key steps in implementing MNR loss during the training phase using the SCL. The process begins with the encoding phase, where input samples are passed through the encoder to generate latent representations. In the pooling phase, these representations are aggregated (e.g., via mean or max pooling) to form fixed-length vectors. Finally, the distance measurement is performed between the pooled anchor-positive pairs and anchor-negative pairs.

Algorithm 1: Pseudocode of implementation MNR Loss

```
Input: Datasets of trade transactions consist of tokenized and batched pairwise anchor
and positive samples
Output: Trained SBERT
Procedure to train SBERT with MNR Loss:
BEGIN
For i = 1 to num_epochs do
    For batch in_batches do
        # extract token embeddings from each pairwise transaction in the batch
        anchor embeddings = extracted encoded anchor samples from transformer-based model
        positive embeddings = extracted encoded positive samples from transformer-based
        model
        # generate pooled vector
        pooled anchor = pooling function (anchor embeddings)
        pooled_positive = pooling function (positive embeddings)
        # similarity consists of n x n vectors
        Similarity = cosine_similarity (pooled_anchor, pooled_positive)
        Score = Similarity * Scale
        # label consists of n x n vectors with I true label and negative labels elsewhere
        in each row
        Loss = CrossEntropyLoss (Score * Label)
        # calculate gradients and update the network
    End
End
END
```

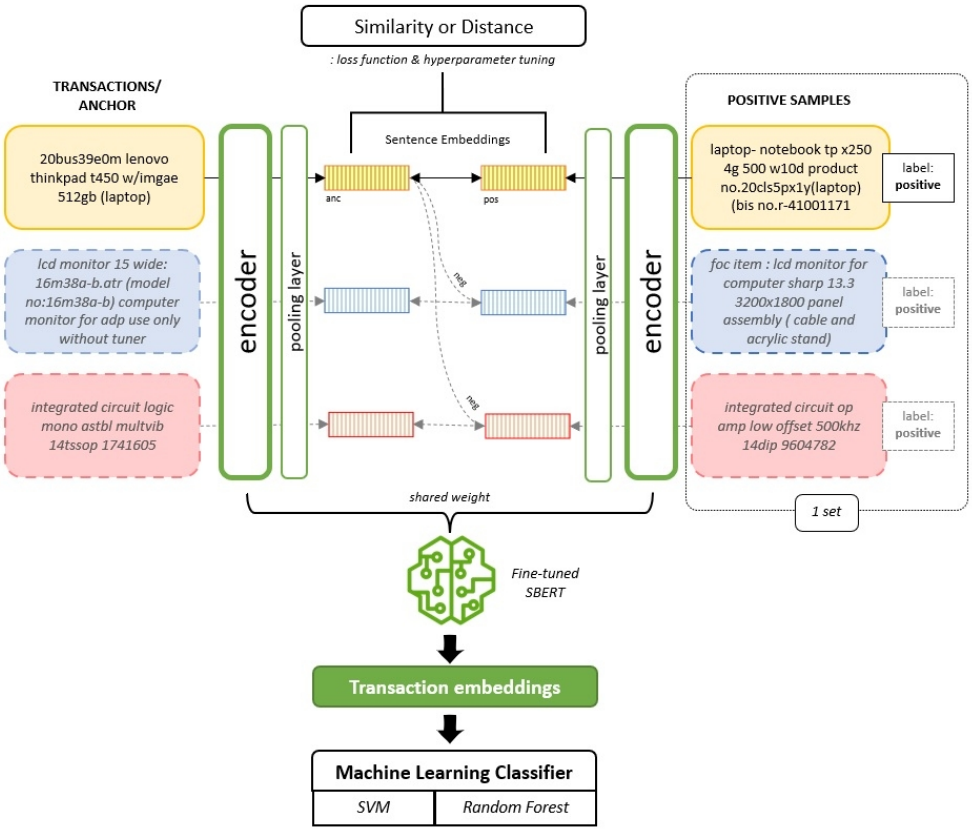

Figure 1. The proposed supervised contrastive learning process using the SBERT framework is illustrated. The transformer-model encodes product description, which is then passed through a pooling layer. During the training phase, anchor samples are paired with corresponding positive samples (from the same class). The model learns to minimise the distance between these anchor-positive pairs in the latent space while maximising the distance between the anchor and samples from other classes (negative samples). Finally, the fine-tuned model is employed to generate transaction embeddings, which are then utilised in a downstream classification task. Source: Author's own work.

251x214mm (96 x 96 DPI)

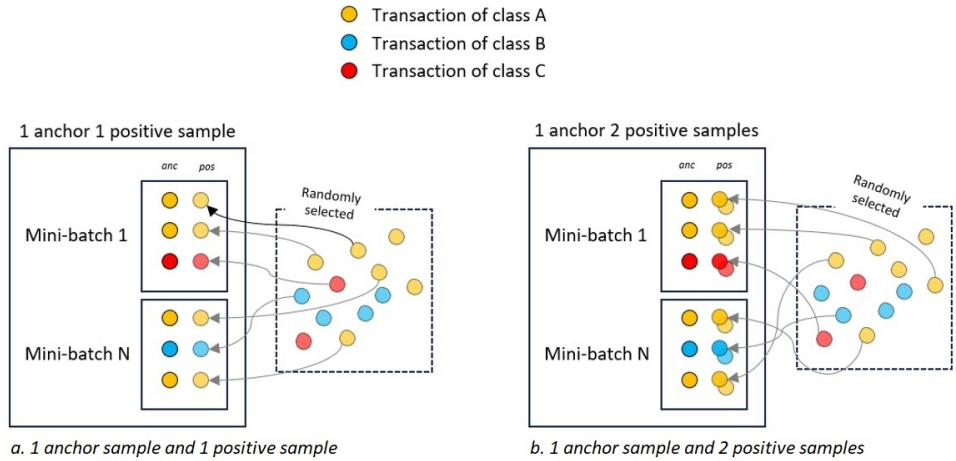

Figure 2. The figure demonstrates the pairwise sample construction process. For each anchor sample, a positive sample is randomly selected from the dataset, all belonging to the same class as the anchor (left). However, an anchor can have more than one positive samples, allowing the model to learn from diverse representations within the same class (right). Source: Author's own work.

316x154mm (96 x 96 DPI)

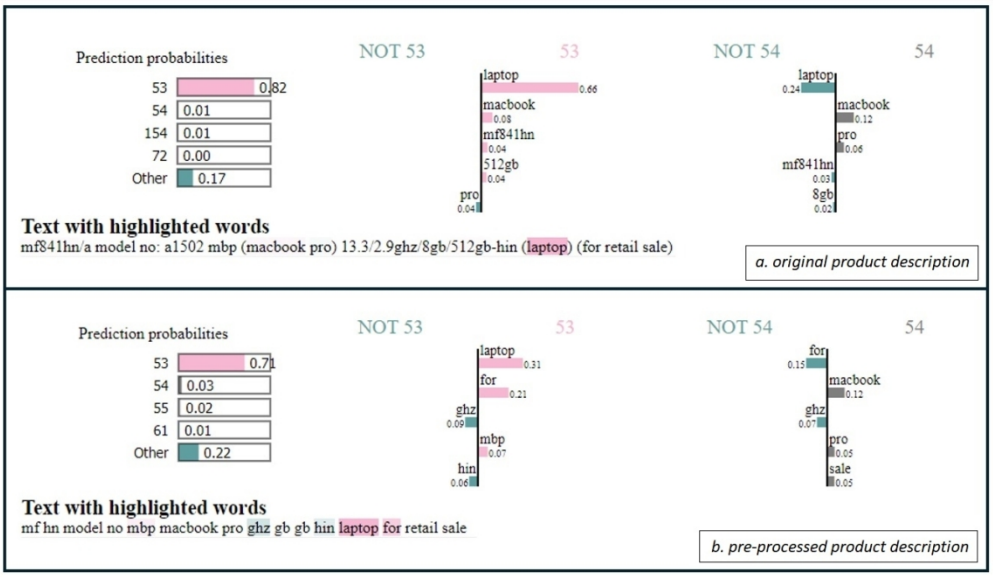

Figure 3. This figure compares the LIME visualisations of model predictions using different data pre-processing techniques. The visualisations indicate higher prediction probabilities for the original text (a), suggesting that the model's confidence in its predictions decreases after certain preprocessing steps (b). This observation implies that retaining specific textual information during preprocessing may enhance prediction accuracy.

354x204mm (96 x 96 DPI)

Detailed description of contents	Custom duty (€)	Tax (€)	Qty	Net weight (kg)	Unit Value (€)	Total value (€)	HS tariff Nr.	Country of origin
Lafuma backpack for men hiking or skiing Volume: 40 L - Dim L: 100 cm - 50 cm	0.00	0.00	1	1.00	100	100	420292	FR
			Total Value	1.00	100	100	Shipping Value: 29.42 (€)	

Service required

☐ Gift ☒ Commercial Sample ☐ Commercial Sample

☐ Document ☐ Return ☐ Other (explain)

Insurance value : 0.00 €

Office of origin / date of posting

GENNEVILLIERS PFC - 10/06/2020

375x93mm (96 x 96 DPI)

Algorithm 1: Pseudocode of implementation MNR Loss

```
Input: Datasets of trade transactions consist of tokenized and batched pairwise anchor and positive samples
Output: Trained SBERT
Procedure to train SBERT with MNR Loss:
BEGIN
For i = 1 to num_epochs do
   For batch in_batches do
      # extract token embeddings from each pairwise transaction in the batch
      anchor embeddings = extracted encoded anchor samples from transformer-based model
      positive embeddings = extracted encoded positive samples from transformer-based model
      # generate pooled vector
      pooled anchor = pooling function (anchor embeddings)
      pooled_positive = pooling function (positive embeddings)
      # similarity consists of n x n vectors
      Similarity = cosine_similarity (pooled_anchor, pooled_positive)
      Score = Similarity * Scale
      # label consists of n x n vectors with I true label and negative labels elsewhere in each row
      Loss = CrossEntropyLoss (Score * Label)
      # calculate gradients and update the network
   End
End
END
```

320x200mm (96 x 96 DPI)

Datasets	Number of Classes	Number of Transactions
Datasets 1 (India commodity trade transactions)	172	66,522
Datasets 2 (U.S. commodity trade transactions)	112	58,003

Table I. The number of transactions and the corresponding commodity classes used in this study for the India and U.S. trade datasets.

HS Code	Commodity Description (Original)	Commodity Descrition (Pre-procecessed)
85423100	INTEGRATED CIRCUIT LOGIC MONO ASTBL MULTVIB 14TSSOP 1741605	INTEGRATED CIRCUIT LOGIC MONO ASTBL MULTVIB TSSOP
85423100	INTEGRATED CIRCUIT OP AMP LOW OFFSET 500KHZ 14DIP 9604782	INTEGRATED CIRCUIT OP AMP LOW OFFSET KHZ DIP
84713010	X2 LAPTOP LENOVO MODEL THINKPAD T430	LAPTOP LENOVO MODEL THINKPAD
84713010	LAPTOP- NOTEBOOK TP X250 4G 500 W10D PRODUCT NO.20CLS5PX1Y(LAPTOP) (BIS NO.R-41001171	LAPTOP NOTEBOOK TP PRODUCT NO CLS PX LAPTOP BIS NO
85285100	LED MONITOR, S22E360H, 21.5, INDIA, LB50/S22ECO, LS22E360HS/XL (22)(60 PCS) (FOR COMPUTER) (SAMSUNG)	LED MONITOR INDIA LB ECO LS HS XL PCS FOR COMPUTER SAMSUNG
84713010	20BUS39E0M LENOVO THINKPAD T450 W/IMGAE 512GB (LAPTOP)	BUS LENOVO THINKPAD IMGAE GB LAPTOP
85285100	LCD MONITOR 15 WIDE: 16M38A-B.ATR (MODEL NO:16M38A-B) COMPUTER MONITOR FOR ADP USE ONLY WITHOUT TUNER	LCD MONITOR WIDE ATR MODEL NO COMPUTER MONITOR FOR ADP USE ONLY WITHOUT TUNER
85285100	FOC ITEM : LCD MONITOR FOR COMPUTER SHARP 13.3 3200X1800 PANEL ASSEMBLY (CABLE AND ACRYLIC STAND)	FOC ITEM LCD MONITOR FOR COMPUTER SHARP PANEL ASSEMBLY CABLE AND ACRYLIC STAND

Table II. The comparison of the original product descriptions with their modified versions after the preprocessing stage. The modifications include removal of irrelevant information, which are commonly applied for the classification models.

Dataset	Method	Cohen Kappa	Macro-average			Micro-average			Weighted average			Model Size
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	
1	Fine-tuning BERT	0.8406	0.8444	0.8426	0.8462	0.8097	0.8426	0.8426	0.8181	0.8426	**0.8406**	418 MB
	Fine-tuning DistilBERT	0.8386	0.8383	0.8407	0.8428	0.8088	0.8407	0.8407	0.8183	0.8407	0.8390	255 MB
	$SBERT_{enc.BERT}$ + SVM	0.8611	0.8700	0.8629	0.8686	0.8370	0.8629	0.8629	0.8464	0.8629	**0.8620**	449 MB
	$SBERT_{enc.DistilBERT}$ + SVM	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603	291 MB
	$SBERT_{enc.BERT}$ + RF	0.8647	0.8691	0.8664	0.8711	0.8445	0.8664	0.8664	0.8512	0.8664	**0.8656**	631 MB
	$SBERT_{enc.DistilBERT}$ + RF	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637	511 MB
2	Fine-tuning BERT	0.7680	0.7508	0.7711	0.7750	0.7127	0.7711	0.7711	0.7210	0.7711	0.7689	418 MB
	Fine-tuning DistilBERT	0.7626	0.7398	0.7658	0.7677	0.7032	0.7658	0.7658	0.7130	0.7658	0.7630	255 MB
	$SBERT_{enc.BERT}$ + SVM	0.7982	0.7881	0.8009	0.8090	0.7603	0.8009	0.8009	0.7681	0.8009	0.8013	437 MB
	$SBERT_{enc.DistilBERT}$ + SVM	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953	278 MB
	$SBERT_{enc.BERT}$ + RF	0.7972	0.7846	0.7998	0.8077	0.7624	0.7998	0.7998	0.7681	0.7998	0.8005	512 MB
	$SBERT_{enc.DistilBERT}$ + RF	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958	371 MB

Table III. The metric highlights the improved performance between baseline models (Fine-tuned BERT and DistilBERT) and fine-tuned SBERT combined with SVM and Random Forest classifiers. Key metrics such as Cohen's Kappa, along with macro, micro, and weighted averages of precision, recall, and F1 score, demonstrate that the fine-tuned SBERT-based models combining with SVM and Random Forest outperform the baselines in terms of classification performance.

Dataset	Method	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	Pre-processed data	0.7803	0.804	0.7831	0.7923	0.7402	0.7831	0.7831	0.757	0.7831	**0.7790**
	Original data	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	**0.8603**
	Random Forest										
	Pre-processed data	0.8007	0.8225	0.8033	0.8121	0.7690	0.8033	0.8033	0.7848	0.8033	**0.8017**
	Original data	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	**0.8637**
2	***SVM***										
	Pre-processed data	0.7143	0.7196	0.7183	0.7338	0.6621	0.7183	0.7183	0.6795	0.7183	**0.7185**
	Original data	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	**0.7953**
	Random Forest										
	Pre-processed data	0.7165	0.7119	0.7203	0.7318	0.665	0.7203	0.7203	0.6788	0.7203	**0.7203**
	Original data	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	**0.7958**

Table IV. The model performance across different data pre-processing techniques. Key metrics such as weighted average precision, recall, and F1-score demonstrate how different pre-processing methods impact classification accuracy.

Dataset	Samples	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	1 anchor, 1 positive	0.8365	0.8485	0.8386	0.8458	0.7965	0.8386	0.8386	0.8105	0.8386	0.8360
	1 anchor, 3 positive	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Random Forest										
	1 anchor, 1 positive	0.8412	0.8552	0.8432	0.8492	0.8051	0.8432	0.8432	0.8200	0.8432	0.8410
	1 anchor, 3 positive	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
2	***SVM***										
	1 anchor, 1 positive	0.7706	0.7621	0.7738	0.7847	0.7165	0.7738	0.7738	0.7304	0.7738	0.7735
	1 anchor, 3 positive	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Random Forest										
	1 anchor, 1 positive	0.7668	0.7517	0.7699	0.7770	0.7137	0.7699	0.7699	0.7244	0.7699	0.7688
	1 anchor, 3 positive	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958

Table V. The table compares the impact of using different numbers of positive samples per anchor on model performance. Some metrics are used to demonstrate how increasing the number of positive samples improves classification accuracy across two datasets.

Dataset	Pooling Strategy	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	CLS	0.8594	0.8696	0.8612	0.8675	0.8300	0.8612	0.8612	0.8425	0.8612	0.8603
	Mean	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Max	0.8564	0.8637	0.8582	0.8645	0.8269	0.8582	0.8582	0.8377	0.8582	0.8574
	Random Forest										
	CLS	0.8611	0.8676	0.8629	0.8686	0.8359	0.8629	0.8629	0.8453	0.8629	0.8621
	Mean	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Max	0.8593	0.8618	0.8611	0.8672	0.835	0.8611	0.8611	0.8414	0.8611	0.8605
2	***SVM***										
	CLS	0.7963	0.7991	0.7990	0.8114	0.7555	0.7990	0.7990	0.7690	0.7990	0.8006
	Mean	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Max	0.7927	0.7866	0.7955	0.8068	0.7529	0.7955	0.7955	0.7629	0.7955	0.7966
	Random Forest										
	CLS	0.7957	0.7857	0.7984	0.8084	0.7561	0.7984	0.7984	0.7654	0.7984	0.7997
	Mean	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Max	0.7909	0.7812	0.7937	0.8038	0.7513	0.7937	0.7937	0.7594	0.7937	0.7945

Table VI. The table compares model performance using various data pooling strategies, including CLS token, MEAN pooling, and MAX pooling. The results show that all three strategies yield similar or comparable performance across key metrics.

Dataset	Loss Function Parameters	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Symmetric MNR Loss	0.8609	0.8722	0.8627	0.8686	0.8342	0.8627	0.8627	0.8455	0.8627	0.8616
	Scale - 50	0.8579	0.8679	0.8597	0.8661	0.8268	0.8597	0.8597	0.8388	0.8597	0.8585
	Dot Product	0.8512	0.8624	0.8531	0.8585	0.8135	0.8531	0.8531	0.8278	0.8531	0.8508
	Random Forest										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Symmetric MNR Loss	0.8638	0.8702	0.8655	0.8707	0.8389	0.8655	0.8655	0.8478	0.8655	0.8645
	Scale - 50	0.8612	0.8687	0.8630	0.8688	0.8340	0.8630	0.8630	0.8440	0.8630	0.8620
	Dot Product	0.8569	0.8675	0.8587	0.8641	0.8279	0.8587	0.8587	0.8397	0.8587	0.8574
2	***SVM***										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Symmetric MNR Loss	0.7968	0.787	0.7996	0.809	0.7531	0.7996	0.7996	0.7638	0.7996	0.8002
	Scale - 50	0.7905	0.7906	0.7933	0.8056	0.7469	0.7933	0.7933	0.7606	0.7933	0.7947
	Dot Product	0.7911	0.7863	0.7939	0.8084	0.7392	0.7939	0.7939	0.7533	0.7939	0.7954
	Random Forest										
	MNR Loss - *(Scale:20, Cosine Similarity)*	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Symmetric MNR Loss	0.7957	0.7782	0.7984	0.8068	0.7546	0.7984	0.7984	0.7606	0.7984	0.7989
	Scale - 50	0.7906	0.7856	0.7934	0.804	0.7518	0.7934	0.7934	0.7618	0.7934	0.7944
	Dot Product	0.7945	0.7875	0.7973	0.8057	0.7519	0.7973	0.7973	0.7617	0.7973	0.7974

Table VII. The table presents the performance comparison of the model with various configurations of the loss function. Key metrics are evaluated, showing how different loss functions settings impact model accuracy.

Dataset	Hyperparameters Training	Cohen Kappa	Macro-average			Micro-average			Weighted average		
			Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	***SVM***										
	Batch 8	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Batch 32	0.8619	0.8805	0.8637	0.8699	0.8359	0.8637	0.8637	0.8504	0.8637	0.8626
	Epoch 10	0.8411	0.8523	0.8432	0.8499	0.7980	0.8432	0.8432	0.8129	0.8432	0.8402
	Epoch 30	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Learning Rate 2e-5	0.8595	0.8717	0.8613	0.8667	0.8335	0.8613	0.8613	0.8457	0.8613	0.8603
	Learning Rate 5e-5	0.8488	0.8554	0.8507	0.8557	0.8228	0.8507	0.8507	0.8326	0.8507	0.8496
	Random Forest										
	Batch 8	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Batch 32	0.8667	0.8834	0.8684	0.8736	0.8416	0.8684	0.8684	0.8554	0.8684	0.8673
	Epoch 10	0.8475	0.8525	0.8494	0.8547	0.8136	0.8494	0.8494	0.8242	0.8494	0.8476
	Epoch 30	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Learning Rate 2e-5	0.8628	0.8707	0.8646	0.8702	0.8403	0.8646	0.8646	0.8486	0.8646	0.8637
	Learning Rate 5e-5	0.8471	0.8515	0.8491	0.8541	0.8213	0.8491	0.8491	0.8299	0.8491	0.8481
2	***SVM***										
	Batch 8	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Batch 32	0.7937	0.8017	0.7964	0.8093	0.7513	0.7964	0.7964	0.7670	0.7964	0.7977
	Epoch 10	0.7779	0.7669	0.7809	0.7916	0.7189	0.7809	0.7809	0.7320	0.7809	0.7795
	Epoch 30	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Learning Rate 2e-5	0.7917	0.7807	0.7945	0.8056	0.7463	0.7945	0.7945	0.7562	0.7945	0.7953
	Learning Rate 5e-5	0.7886	0.7824	0.7914	0.8026	0.7510	0.7914	0.7914	0.7603	0.7914	0.7928
	Random Forest										
	Batch 8	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Batch 32	0.7939	0.7864	0.7967	0.8062	0.7506	0.7967	0.7967	0.7619	0.7967	0.7972
	Epoch 10	0.7813	0.7658	0.7842	0.7922	0.7321	0.7842	0.7842	0.7418	0.7842	0.7839
	Epoch 30	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Learning Rate 2e-5	0.7924	0.7771	0.7952	0.8045	0.7498	0.7952	0.7952	0.7565	0.7952	0.7958
	Learning Rate 5e-5	0.7854	0.7716	0.7882	0.7974	0.7473	0.7882	0.7882	0.7536	0.7882	0.7891

Table VIII. The table presents the impact of various hyperparameter configurations on model performance. Key metrics are employed to assess the effectiveness of different learning rates, batch sizes, and epochs. The results show that optimal hyperparameter tuning can significantly improve the model's classification accuracy and overall performance.