TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Deniss Ruder 172659IVSM

Application of Machine Learning for Automated HS-6 Code Assignment

Master's thesis

Supervisor: Margarita Spitšakova
PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Deniss Ruder 172659IVSM

Masinõppe meetodid HS-6 koodide automatiseeritud identifitseerimiseks

Magistritöö

Juhendaja: Margarita Spitšakova
Doktorikraad

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature, and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Deniss Ruder

03.08.2020

Abstract

In recent years, the European Union attempts to reduce losses from unpaid VAT and customs duties caused by the harmonized system code misclassification. The current study is designed to explore machine learning solutions for this problem by classifying six-digit HS codes required in the EU according to their textual cargo descriptions. Our research examined 1124874 US import cargo descriptions represented by 3243 unique HS-6 codes applying Rocchio, MLR, MNB, k-NN, Decision tree, Random forest, SVM, DNN, and CNN classifiers with TF-IDF, Word2Vec, Doc2Vec, and GloVe extracted feature vectors. DNN model with TF-IDF weights found by the RMDL achieved the highest weighted accuracy of 61%. The received results indicate that machine learning algorithms are efficient for assigning HS-6 level codes, even according to uninformative descriptions.

This thesis is written in English and is 49 pages long, including 7 chapters, 19 figures and 23 tables.

Annotatsioon

Viimastel aastatel püütakse Euroopa Liidus vähendada käibemaksu jatollimaksu tasumata jätmist, mis tuleneb kaupade valeklassifitseerimisega harmoneeritud süsteemis. Käesoleva uuringu eesmärk on leida masinõppe meetodeid, mis sobiks kaupade klassifitseerimiseks kasutades nende tekstilisi kirjeldusi. Magistristöös vaadeldi andme hulka, kus on 1124874 USA-st imporditud kauba kirjeldust, mis esindavad 3243 ainulaadset HS-6 koodi. Laheduses kasutati Rocchio, MLR, MNB, k-NN, otsustuspuude, juhuslike metsade, tugivektormasinate, DNN ja CNN põhiseid klassifikaatoreid koos TF-IDF, Word2Vec, Doc2Vec ja GloVe tunnusvektori süsteemiga. RMDL-i leitud TFN-IDF-i kaaludega DNN-mudelsaavutas suurimat kaalutud täpsust 61% ulatuses. Eksperimentide tulemused näitavad, et masinõpe algoritmid on võimelised HS-6 tasemekoode tuvastama isegi mitteinformatiivsetest kirjeldustest.

Lõputöö on kirjutatud inglises keeles ning sisaldab teksti 49 leheküljel, 7 peatükki, 19 joonist ja 23 tabelit.

Table of contents

1 Introduction

1.1 Motivation

Both businesses and customers from the European Union (EU) are actively using e-commerce, ordering products on the internet. According to 2019, residents of the European Union spent 621€ billion on online sales [1]. In 2020, German online statistics agency Statista showed that during the COVID-19 pandemic, the number of online retail markets was growing and expected to expand even further [2]. To survive, even the smallest and inactive companies urged to go online. This additional growth, as well as e-commerce laws imperfections, force the EU to take action adequately handling the processing of goods within the EU and from abroad.

All the commodities, entering the EU or any Member States, taxed by Value Added Tax (VAT), a general consumption tax. When an EU-based company sends its goods to consumers of another Member State, it either applies its local VAT (up to certain threshold) or VAT of the destination country [3]. Non-EU based sellers, on the other hand, have to pay VAT in the recipient country and are required to go through customs clearance. However, low-value goods (150€ or less) imported from outside of the EU are not subject to customs expenses, and packages worth 22€ or less (depending on the state), are fully exempted from VAT.

These reliefs create an opportunity for fraudulent non-EU traders to abuse the system and avoid tax payment. According to recent studies [4], as a result of this politics, VAT is levied only on 35% of all EU imported cargo. In 2019 the Court of Auditors conducted an audit [5] and estimated losses on supplies of low-value imports from non-EU countries around 5 billion annually. The losses from the collection of customs duties were approximated to 0.25 billion per year.

To mitigate the damage, the European Commission proposed the EU's "e-commerce package" [6], which comes into force on the 1st of July 2021. Among other provisions, this package introduces a complete rejection of the current VAT exemption rule for commodities cheaper than 22€. Another significant change is that the "Mini One Stop Shop" (MOSS) system, previously used for VAT payment for electronic services, is extending its scope to e-commerce, becoming "One Stop Shop" (OSS) [7]. When the package comes into force, both EU and non-EU traders will be obliged to register VAT payments and customs clearance costs inside the OSS. The purpose of these measures is to identify fraudulent sellers and build a unified taxation system convenient for all parties.

Discovering fraud behavior is a complex technical task that is first related to determining the original price of the goods, and accordingly, the amount of levied tax. In its turn, the price can be defined only by the type of the product, or more specifically, its identification number called the Harmonized System (HS) code. The HS code assignment is often subject to errors, which leads to product misclassification and the possibility of fraud. Thus, to accurately determine the price, customs duties, VAT, and fraud attempts, the OSS has to solve the problem of correct HS code categorization.

1.2 Problem

To get competitive advantages by reducing the final price for the consumers, sellers consciously hide the real value and category of the goods. Such sellers exploit the isolation of external trading systems from the EU customs that do not directly access this information. As a result, the data obtained by the customs are often inconsistent with the information on e-commerce platforms and real transaction details between the sellers and buyers. Nevertheless, the misclassification happens not only due to the fraud attempts. The product information loses its original state by going through several processing stages: from the seller to the sender, the shipper, and the carrier, that usually manually assign HS codes. The manual code assignment is a very error-prone task that requires following the World Customs Organization (WCO) guidelines and overall workers' high expertise, impossible with this number of stakeholders, and the volume of e-commerce.

To solve the misclassification problem, we need to understand what form the information is dispatched to the customs. The acquired data usually include details about contacts, carriage, contracts, voyage, and other cargo-related information. Besides other fields, it contains HS numbers, and special remarks called cargo descriptions.

Harmonized System code: HS code is a unique identification code given to each category of the products. Developed and maintained by WCO, HS codes are used by customs authorities to collect duties and taxes in more than 200 countries [8]. The WCO-defined version of the system contains six digits, having sections, chapters, headings, and subheadings in its structure (Figure 1). All imported commodities into the EU are required to present at least this basic six-digit version of the code (HS-6).

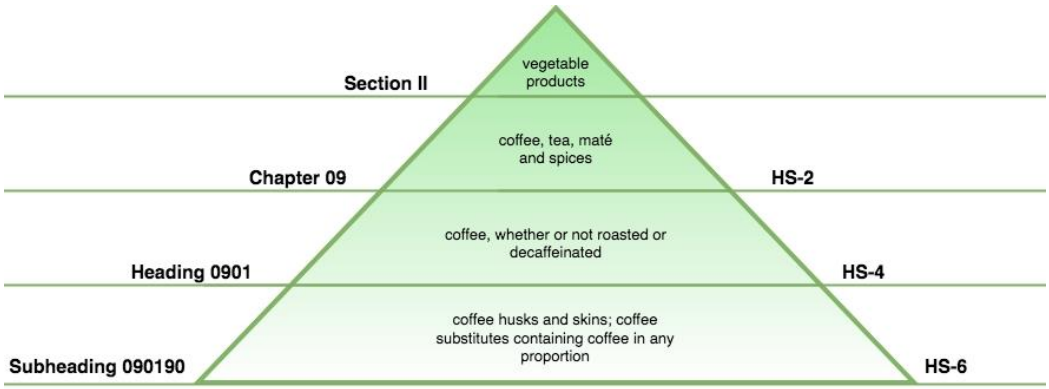

Figure 1. The example of HS code structure

The sellers often do not explicitly indicate the HS-6 when sending their goods, providing shorter codes, or merely generic product descriptions. At times, they are missing completely, and the customs, to identify the HS code, contact customers by sending them declarations (e.g., it is common in Estonia). Thus, HS codes reaching the customs are often either absent or incorrectly classified.

Cargo descriptions: We suggest determining the real product codes by categorizing them based on the cargo descriptions. Cargo descriptions are special commentaries made by the sender, the shipper, and the carrier while processing the bookings. In addition to commodity descriptions, these notes usually contain a lot of product-unrelated data such as payment, shipment, or delivery details.

The first naive intuition of how to link cargo descriptions to their codes would be a database search. However, there are several reasons why a simple database query does not give any meaningful results when trying to match the cargo description to its product code: (1) the harmonized system complexity, (2) the necessity to follow guidelines, (3) the continuous nomenclature revisions, (4) the gap in vocabulary, and (5) the text noisiness.

1. **The harmonized system complexity**: The HS system is complex; it contains approximately 5,300 commodity groups, 99 chapters, 21 sections, and an infinite number of local extensions.
2. **The necessity to follow guidelines**: For the proper use of the system, it is necessary to apply recommendations and clarifications stated by the WCO. Unfortunately, neither the seller nor the carrier follows these rules creating an ambiguity.
3. **The continuous nomenclature revisions**: HS code nomenclature is not static. The WCO regulates its development and maintenance, making amendments every 5-6 years [8], which are then adopted for customs regulations by local governments. National authorities also continually refine, revise, and manage their extended versions of the system. This pace of change requires a classification approach to be flexible to any updates in the harmonized system.
4. **The gap in vocabulary**: There is a considerable gap between goods text descriptions and their HS system counterparts. For instance, Apple Ipad may be classified as 8471.41 “Automatic data-processing machines and units thereof; magnetic or optical readers, machines for transcribing data onto data media in coded form and machines for processing such data, not elsewhere specified or included.”. This definition has almost no relevant connection with the tablet’s text description provided by sellers.
5. **The text noisiness**: Text descriptions contain too much unnecessary information, such as transportation comments, brands, company names, typos, materials, color, and other keywords not related to the product’s class.

Considering the listed points, the HS misclassification problem requires a different answer than a database string search. The solution proposed in our research is to use machine learning (ML) models, which in recent years became the primary tool for solving text classification problems. We intend to examine some of the popular ML classification algorithms and confirm or disapprove their practical application for HS-6 categorization.

Our work is exclusively interested in the HS-6, as it is the minimal prerequisite for imported goods into the EU, and because the existing studies only considered cases with chapter and heading level codes. The HS-2 and HS-4 codes contain significantly fewer classes, less detailed product definitions, and are overall better classified by short keywords extracted from cargo descriptions.

Thus, the product misclassification problem is caused by intentional fraud attempts and accidental errors in code assignment. We propose to classify HS-6 level codes, applied in the EU, according to cargo descriptions, special commentaries made at the shipping process. Due to the noisiness of cargo descriptions, the terminology inconsistencies, and the problems related to the domain of the harmonized system, misclassification cannot be solved by a simple database search. Therefore, our thesis suggests predicting product codes by using ML text classification models.

We conduct our study in five different parts: (1) Background and related work, (2) Methodology, (3) Results, (4) Discussion, and (5) Summary.

1. **Background and related work**: This chapter introduces the reader to the text classification systems domain, overviews the existing studies related to HS code classification, and contains references to all relevant literature.
2. **Methodology**: This chapter presents and analyzes the dataset, prepares it for the feature extraction and classification steps, sets the evaluation criteria, and conducts all related experiments.
3. **Results**: This chapter contains the resulting performance table of examined classifiers and feature extractors.
4. **Discussion**: This chapter sums all the findings of the research, compares and interprets applied methods, validates the results of the classification.
5. **Summary**: The final part concludes the research and proposes further work.

2 Background and related work

2.1 Background

Significant development of machine learning algorithms, natural language processing (NLP), and text mining led to the broad application of text classification in many domains. It is utilized in recommender systems, document summarization, information filtering, sentiment analysis, knowledge management, and other fields. In this section, we research how text classification is accomplished and investigate algorithms more suitable for classifying HS-6 codes. This section is based on a state-of-the-art survey [9] on text classification algorithms conducted in 2019.

The text classification is a pipeline of operations consistently applied to the input text document. In general, the initial raw document is split into sequences of text as $D = \{X_1, X_2, ..., X_N\}$, where X_i denotes a data point that consists of s sentences, each containing w_s words with l_w letters [10]. Text sequence may refer to a different scope of the text: whole document, paragraph, sentence, or sub-sentence levels. Every data point then binds to a class value index from an array of k classes. For instance, in our case, each product description represents a complete text segment that has to be labeled with the set of all possible HS-6 codes.

The typical text classification system pipeline contains five stages (Figure 2): (1) Data pre-processing, (2) Feature extraction, (3) Dimensionality reduction, (4) Classification, and (5) Evaluation.

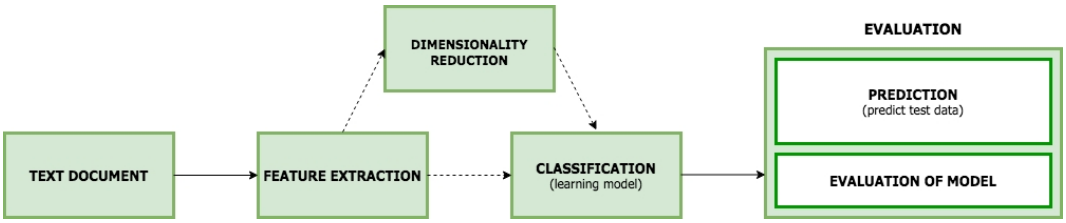

Figure 2. Text classification system pipeline

1. **Data pre-processing**: Text sequences are pre-cleansed by removing redundant characters or words. The cleansing can help eliminate punctuation marks, slang, transportation notes, brands, and unnecessary keywords, e.g., color or goods characteristics. Data cleansing is performed through Tokenization [11][12], Capitalization [13][14], Noise Removal [15], Stemming [16], Spelling Correction [17][18], Lemmatization [19][20], and other techniques. The data splitting, stratified sampling, and rebalancing may also be referred to the pre-processing phase.
2. **Feature extraction**: At this stage, pre-processed text sequences are converted into a feature space needed for the classification step. Feature extraction is generally achieved using two methods: word embedding (e.g., Word2Vec [21], Doc2Vec [22], GloVe [23], FastText [24][25]), or weighted word techniques (e.g., Term Frequency-Inverse Document Frequency (TF-IDF), Term Frequency (TF) [26]).

3. **Dimensionality reduction (Optional)**: Reducing the size of feature space increases the performance of the system. Decreasing complexity might be useful when there are too many unique words or too few pre-processing phases. Most common approaches are Principal Component Analysis (PCA) [27], Linear Discriminant Analysis (LDA) [28][29], and non-negative matrix factorization (NMF) [30] [31].
4. **Classification**: After being pre-processed, extracted into feature space, and optionally reduced, the data is taken for the classification. Choosing the right classification method is the most critical step in text categorization. Commonly used multi-class techniques are standard algorithms (Rocchio classification [32], Multinomial Logistic Regression (MLR) classifier [33][34], Multinomial Naïve Bayes (MNB) classifier [35]), non-parametric (k-nearest neighbors (k-NN) [36] [37], Support Vector Machine (SVM) [38][39]), tree-based (Decision trees [40], and Random Forests [41]), and deep learning neural networks [42] (e.g., convolutional neural network (CNN) [43], deep neural network (DNN) [44], or recurrent neural network (RNN) [45]). The deep learning algorithms are usually more precise than standard models but more complicated to understand and implement. Figure 3 gives a rough comparison of the complexity and accuracy of both types of algorithms:

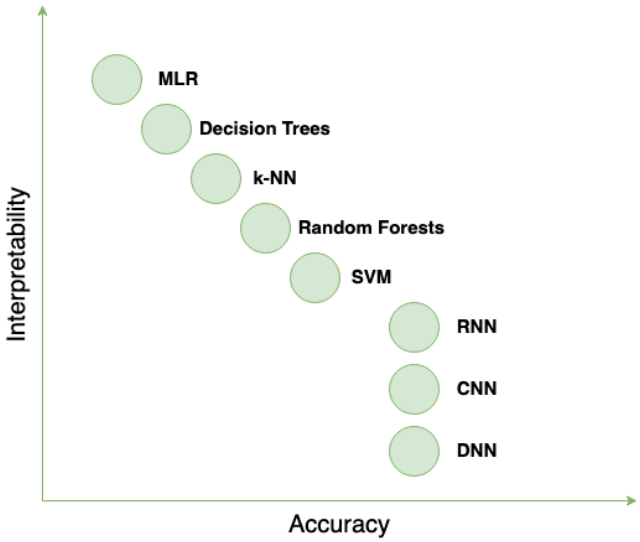

Figure 3. The interpretability and accuracy comparison between linear and deep learning models.

5. **Evaluation**: The final phase of the classification system is evaluating predictions. It lets us assess the effectiveness of algorithms and compare them to find the best possible results. Most frequently used metrics are accuracy [46], F-1 Score [47], Matthews Correlation Coefficient (MCC) [48], receiver operating characteristics (ROC) [49], and area under the ROC curve (AUC) [50].

Thus, the obligatory pipeline structure of the common text classification system involves the cleansing, feature extraction, classification, and evaluation steps, which design is also a baseline for our methodology chapter.

2.2 Related work

There are relatively few researches on the HS code categorization topic, especially it concerns classifying HS-6 level codes and covering full nomenclature. To be specific, we did not find any studies dedicated exclusively to HS-6 codes, despite being the minimum requirement for many countries. Most of the existing researches explored only HS-2, HS-4, or separate chapters. We believe that such a situation is due to the high demand in the commercial sphere, and in practice, HS-6 text classification systems are actually widely applied. Nevertheless, the work present in this section is highly valuable and used as a basis for our thesis.

In 2015, researchers from the National University of Singapore published a paper [51] that suggested using the Background Net approach for the classifier and vector space model for feature extraction. The authors used the data provided by Singaporean Customs, which is not publicly available for the analysis. Data preparation steps included deletion of punctuations, content in brackets, capitalization, and duplicates. The research focused on classifying goods for Chapter 22 and Chapter 90 without stating the classified code length. Their results showed that, in some instances, the B-Net delivered encouraging 90% accuracy. Unfortunately, despite the research value, the paper lacks details, and authors admit that their approach wrongly classified 84.14% of short record descriptions and encountered issues with high-level descriptions.

In 2018, researchers from Brazil conducted a study [52] written in Portuguese on classifying NCM codes (Brazilian HS code extension) for the same chapters 22 and 90. The data researched in a paper was collected from Brazil's government and is not publicly available for exploration. The authors proposed the application of the NBC classification technique with a TF-IDF feature extractor. Data pre-processing included deletion of stop words, tokenization, and stemming. The Naïve Bayes classifier achieved high numbers of accuracy varying from 83% to 98%.

In 2019, researchers from the Beijing Institute of Technology conducted a case study [53] on classifying text descriptions and images from e-commerce websites according to HS codes for boots. By learning 10,000 images, text descriptions, and their parameters such as the height of the shoe shaft or length of the inner soles, they classified them into four HS categories: 64039111, 64039119, 64039191 and 64039199. The pre-processing and feature extraction steps involved using the NLPIR library, the removal of numerals, quantifiers, and punctuation. The researchers proposed building two convolutional neural networks (CNN), a deep learning method, and a fusion algorithm for the classifier. Text CNN showed much higher overall accuracy achieving 93.4%; image CNN results, on the other hand, reached only 76.83%.

All mentioned above papers focused on specific chapters or individual products, and their data is not accessible for the explicit analysis. A recent master's thesis [54] attempted to classify around 1000 HS-2 and HS-4 codes altogether. The author used Enigma's US import open data for text descriptions and HS codes and various accessible product ontologies to obtain well-defined word embeddings. The classification stages included using different models: LR, MNB, k-NN, SVM and CNN with TF-IDF, and Word2Vec as feature extractors. During the experiments, the researcher made a comparison between different conjunctions of classifiers and feature extractors. The analysis demonstrated more than 80% accuracy for all combinations; the CNN classifier showing the best performance, with 82.7% to 92.3% accuracy depending on the feature extractor and data. However, due to the lack of time and other constraints, there could be an enlargement of the number of unique words in a dictionary, more

further data preparation, improved Word2Vec parameters, and more extensive research in general.

Most recent conference proceeding [48] explored more than 22,000,000 Dubai Customs records. Their data cleansing carried standard removal of duplicates, punctuation, non-English words, numbers, and lemmatization. To perform feature extraction, the authors used TF-IDF value-based vectors. Machine learning models applied and compared in the research were: MNB, k-NN, Decision tree, Random Forests, SVM, and Adaboost. Testing and evaluation were conducted on the HS-4 and "entire HS code", most likely to be HS-6 or local HS extension because of the lower prediction results. Linear SVM was once again confirmed to be the best classifier with 75.40% accuracy for the lengthened code. This work, published at the time of writing the thesis, is the first non-HS-4 code prediction study; and to some degree sample of measurement for our results.

Thus, most reviewed studies were conducted on private data impossible to analyze and compare with our results. The higher numbers may indicate less noisiness of the descriptions in comparison with the US import dataset. The only study [54] carried on the US import data except for 2016-2017 classified 1271 unique HS-4 codes, reaching 88% accuracy with CNN. This result looks very promising, considering that it applied less pre-processing steps, and did not utilize the RMDL. For instance, we achieved 57% accuracy using the same Word2Vec Skip-gram and CNN on classifying 3243 HS-6 codes, and 35% accuracy without lemmatization and the removal of non-English words steps missing in the compared research. The possible explanation could be that our results were received with stratified sampling and all class appearances in both train and test sets, hence taking into account the multi-class focus of the task. However, relevant weighted F-1 scores are not available, and we cannot compare the proportional performance of each class. Therefore, despite the significance of the related work, it is hard to compare due to the diverse data, absence of weighted accuracy scores, less descriptive HS codes, and the overall number of classes.

3 Methodology

3.1 Data pre-processing

Data pre-processing is the initial and critical step for the text classification system, as it directly affects the performance of both feature extraction and classification. This chapter introduces the dataset, implements, and describes methods applied for preparing, cleansing and splitting the data, explores and visualizes the results of each step. All listed actions were conducted using Jupyter notebooks and Python open-source libraries: Pandas, Matplotlib, Plotly, Cufflinks, Nltk, and others.

3.1.1 Dataset

US Imports - Automated Manifest System (AMS) Shipments is the Enigma dataset [59] available at Amazon AWS via free subscription. Enigma sources this data from the US Customs and Border Protection (CBP), which gathers and weekly revives all the merchandise entering the country. The data comprises manufacturer, shipper, vessel and carrier details, cargo descriptions, tariff codes, and other fields.

Tariff codes used in the dataset are either common HS-2, HS-4, HS-6, or a lengthened variation (7-10 digits) of the HS system called The Harmonized Tariff Schedule of the United States (HTS). The first six numbers of the HTS have the same definition as the HS-6 and can be successfully applied for our study. Our data preparation step, besides other procedures, is intended to truncate these codes to HS-6 and remove irrelevant HS-2 and HS-4.

Cargo descriptions are remarks made by booking officers to confirm the booking requests. Most of the time, these commentaries are written by non-native English speakers and contain many mistakes, typos, and unrelated information, such as shipment or payment details. The goal of further data cleansing is to eliminate this unnecessary information and keep only words related to product descriptions.

In order to collect as much data as possible, we use all Enigma's available tables dated to 2018-2020 and revised in 29/01/2020, 08/01/2020, and /06/07/2020.

3.1.2 Preparation

The US is the largest importer on earth, and its 2018-2020 import data contains millions of records. However, the dataset requires comprehensive preparation to be cleansed at the upcoming stages. In general, this preparation involves the following steps: merging the data for different periods, excluding unnecessary or corrupt records, and converting it into shape ready for tokenization and normalization.

In our dataset, we are solely interested in cargo descriptions and tariff codes tables that initially hold 124,000,000 and 41,000,000 records. Such a significant difference between the number of entries is described by information redundancy, e.g., when multiple containers hold the same commodities. Thus, most of these records are either duplicates or almost identical descriptions of the same products.

To prepare the data for our needs, we: (1) merge tables for different years, (2) make inner join of two tables, (3) drop unnecessary columns, (4) drop the duplicates, (5) drop missing values, (6) remove corrupted HS codes, (7) remove HS-2 and HS-4, (8) convert HTS to HS-6, (9) resolve the conflict with ambiguous records.

1. **Merge tables for different years**: All Enigma's tables are available in separate revisions, at this step, merged into two big tables: cargo descriptions and tariff codes.
2. **Make an inner join of two tables**: We join tables by the columns of identifier, description sequence number, and container.
3. **Drop unnecessary columns**: The merged tables contain numerous irrelevant columns that are all being discarded, omitting only two columns: harmonized number and description text.
4. **Drop the duplicates**: Duplicated records drastically reduce the performance of the classification and may mislead the actual results.
5. **Drop missing values**: Some records include null values that may turn into errors at further processing stages.
6. **Remove corrupted HS codes**: At the later stages, we recognized that the HS code column contained spaces, punctuation, non-existing codes and other symbols that led to crashes.
7. **Drop HS-2 and HS-4**: We do not consider using HS-2 and HS-4 codes, as HS-6 is the minimal prerequisite for our research.
8. **Convert HTS to HS-6**: As we stated before, HS-6 is the base for all the local extensions, and therefore, their first six digits are the same. To retain more records, we shorten the US HTS-7-10 to HS-6.
9. **Resolve the conflict with ambiguous records**: Despite removed duplicates, particular entries hold the same text descriptions but point to two different HS codes (Table 1). To resolve the issue, we keep only the first occurrences of such records.

HYDRAULIC PUMP PARTS	84122981
HYDRAULIC PUMP PARTS	848389

Table 1. An example of an ambiguous record.

3.1.3 Preparation results

Before applying the next cleansing techniques, we make an analysis of the prepared data. Data exploration allows us to discover additional cleaning methods, specify the number of classes, and collect various statistical information about researched columns.

The data exploration is done in three parts: (1) overall statistics, (2) harmonized number, and (3) description text columns stats.

1. **The overall statistics**: After steps executed in the data preparation, our final table contains only 1,220,834 entries. Table 2 shows the basic statistical details about both of the columns:

	Count	Unique	Top	Frequency
Description text	1,220,834	1,220,834	R1CSUR R2 VPSUB …	1
Harmonized number	1,220,834	4328	731815	39,391

Table 2. Basic statistical information after the preparation

2. **The harmonized number**: Our next goal is to determine how unbalanced our data are. It will directly impact the choice of evaluation metrics and the overall objectivity of our research.

   Table 3 shows statistics on the number of matches for every HS code:

Count	4328.00
Mean	282.08
Standard deviation	1501.22
Minimum	1.00
Maximum	39,391.00

Table 3. The harmonized number column statistics

   Figure 4 shows the frequency distribution of all the HS codes according to their number of entries.

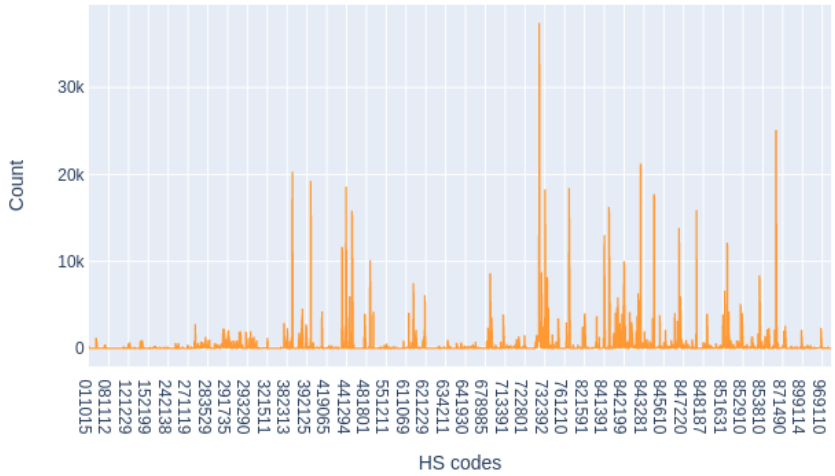

Figure 4. HS code frequency distribution plot

3. **The description text**: The exploration of the description text column helps to compare prepared data with the cleansing results at the following cleansing step. There are several ways of comparing text descriptions: in terms of their (1) overall vocabulary, (2) symbolic length, and (3) word length.

   Vocabulary: In total, there are 20,265,061 words, from which 1,405,333 are unique. The longest word in the vocabulary contains 22 symbols.

   Length in symbols: The statistics about the symbolic length of the descriptions can be viewed in Table 4:

Count	1,220,834.00
Mean	94.11
Standard deviation	61.68
Minimum	1.00
Maximum	367.00

Table 4. The description length statistics in symbols

The frequency distribution histogram of the description length in symbols can be seen in Figure 5:

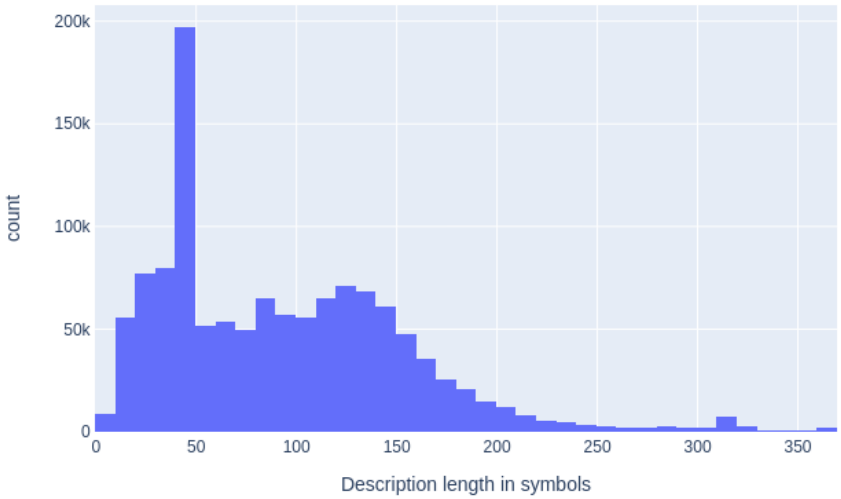

Figure 5. The frequency distribution histogram in symbols

Length in words: Table 5 shows the statistics for description length in words:

Count	1,220,834.00
Mean	16.60
Standard deviation	11.71
Minimum	1.00
Maximum	124.00

Table 5. The description length statistics in words

Figure 6 shows the frequency distribution plot according to words:

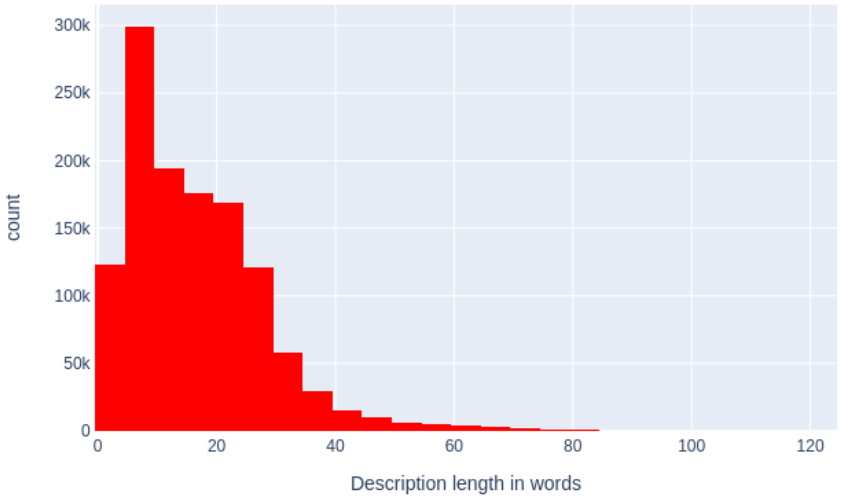

Figure 6. The frequency distribution histogram in words

3.1.4 Cleansing

The cleansing step is aimed to clean the descriptions from the unnecessary text that will not affect the classification performance. It is achieved in two main stages: the tokenization and the normalization.

Tokenization: is a key method for data cleansing, feature extraction, and classifier that is an absolute necessity for any text classification system. Tokenization is a process that splits text sequences into sentences, phrases, words, phrases, characters, or other units, called tokens. In essence, it can also be described as lexical analysis. When applied at the word level, it translates a sentence into a list of words by implementing particular segmentation rules. We use the word tokenizer available in the Natural Language Toolkit (Nltk) library.

Normalization: In data pre-processing, normalization generally refers to a series of similar processing tasks that transform the tokenized text into a specific canonical form. In our work, we approach ten distinct operations on the tokenized text: (1) converting to lower case, (2) removing words with non-ASCII characters, (3) converting textual numbers to digits, (4) removing words with non-alphabetic characters, (5) removing punctuation and stop words, (6) part-of-speech (POS) tagging, (7) lemmatization, (8) removing non-English words, (9) removing noise, and (10) removing empty lists.

1. **Converting to lower case**: In our domain, capitalized letters do not impact the classification. Moreover, ML models are case sensitive, hence lowering the case lessens the vocabulary and simplifies further computations.
2. **Removing words with non-ASCII characters**: This is an optional step that allows us to speed up calculations at the next stages.
3. **Converting textual numbers to digits**: Word numbers such as “one” or “twenty” are converted to digits to be removed at the subsequent step.
4. **Removing words with non-alphabetic characters**: Several types of non-alpha symbols are met in text descriptions: digits, contract numbers, prices, HS codes, container numbers, number of packages, number of pallets, and other quantities. These words do not contribute any significant features to the goods and slow down the training process.
5. **Removing punctuation and stop words**: In our case, punctuation and stop words do not represent any semantic values and still require computing resources. Their removal reduces the number of features and increases the speed of model training.
6. **Part-of-speech tagging**: To support correct lemmatization for different parts of speech, we tag words for adjectives, nouns, verbs, and adjectives by utilizing Nltk corpus.
7. **Lemmatization**: Two main strategies to shorten and unify word form while keeping the same semantic meaning are lemmatization and stemming. Lemmatization replaces a word’s suffix with a different one or eliminates it to get the basic word form called a lemma. Stemming is a more radical method; it reduces inflected words to their word stem without substitutions, hence not implying full morphological analysis. Despite stemming advantages in computational speed, it usually has fewer precision gains than lemmatization. In our work, we use lemmatization in a combination with POS tagging, which allows us to get rid of many useless words at the next cleansing step. The lemmatized results are especially useful for TF-IDF extracted values, which in our research are used by all linear models and DNN. However, lemmatization may affect the performance of CNN that uses word

embeddings as input to its model, as it reduces the text's semantic and syntactic properties. To utilize the benefits of both feature extraction approaches, we keep two separate datasets: for TF-IDF and word embeddings.

8. **Removing non-English words**: To delete abbreviations, incorrect spellings, and non-English words from the list of tokens, we use Nltk WordNet and Unix words corpora. Unix corpus contains approximately 250,000 English words. WordNet provides about 155 000 English words. Both corpora contain only words basic forms, and therefore should be processed after POS and lemmatization.
9. **Removing noise**: After previous procedures, there are still many irrelevant tokens that corrupt our data. To exclude these words, we define a dictionary by searching for the most frequent occurrences within all descriptions. Generally, unnecessary tokens can be divided into three groups: words related to logistics, words related to payments, and random noise. Common examples of transportation comments are related to contacts, packing, shipping, containers, quantity, and weight. Words linked to payments include contracts, freight, duties, invoices, and business parties, i.e., shipper, forwarder, consignee, carrier, or notify. The last group covers country full names and their abbreviations, color, single characters, and other redundant noise. The deletion of determined vocabulary reduced the number of tokens and improved accuracy.
10. **Removing empty lists of tokens**: Some records did not contain words associated with the product information, and therefore, turned into empty lists. Our final pre-processing iteration eliminates these rows.

3.1.5 Cleansing results

In this section, we review the cleansing outcomes and their difference from the initial preparation in terms of the amount of unique HS codes, the symbolic and word length of descriptions, and its vocabulary. The data analysis follows the same structure as in the data exploration section; we describe: (1) overall statistics, (2) harmonized number, and (3) description text columns statistics.

1. **The overall statistics**: After applying preceding procedures, the table held 1,125,750 entries. Consequently, the table size in memory reduced more than three times, from 191,6 MB to 60.2 MB. Table 6 shows overall changes in statistical details after the cleansing:

	Count	Unique	Top	Frequency
Description Text	1,125,750	405,941	screw	8236
Harmonized Number	1,125,750	4119	731815	37,437

Table 6. Basic statistical information after the cleansing

2. **The harmonized number**: The removal of empty lists resulted in a slight decrease in the number of classes; there remained 4119 product categories.

3. **The description text**: For the fair comparison with the preparation step, we removed all duplicated tokens, sorted them alphabetically, and joined to a single space-separated string. Such a representation illustrates how many descriptions are actually the same, despite having different order and inner duplicates. As in the exploration section, we make a comparison of: (1) vocabulary, (2) length in symbols, and (3) length in words.

   Vocabulary: Total number of words dropped from 20,265,061 to 5,158,174, unique words shrunk from 1,405,333 to 17,476. The longest word in a dictionary still contains 22 characters.

   Length in symbols: Table 7 compares the symbolic length statistics at both steps:

	The preparation	The cleansing
Count	1,220,834.00	1,125,750.00
Mean	94.11	25.33
Standard deviation	61.68	18.54
Minimum	1.00	2.00
Maximum	367.00	267.00

Table 7. The symbolic description length statistics comparison

   A modified frequency distribution histogram of the symbolic length of the descriptions can be observed in Figure 7:

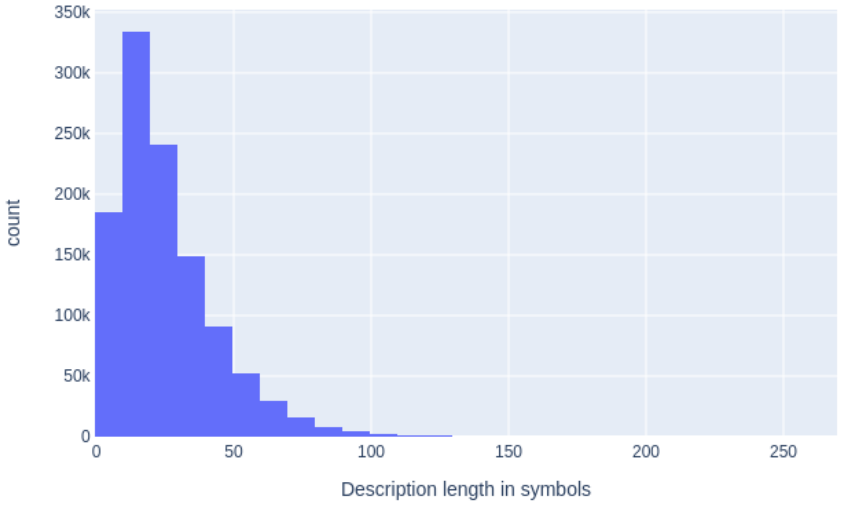

Figure 7. The symbolic frequency distribution histogram after the cleansing

   Length in words: Table 8 shows the statistics for description length in words:

	The preparation	The cleansing
Count	1,220,834.00	1,125,750.00
Mean	16.60	4.01
Standard deviation	11.71	2.76
Minimum	1.00	1.00
Maximum	124.00	45.00

Table 8. The word description length statistics comparison

Figure 8 shows the word frequency distribution histogram after the applied cleansing.

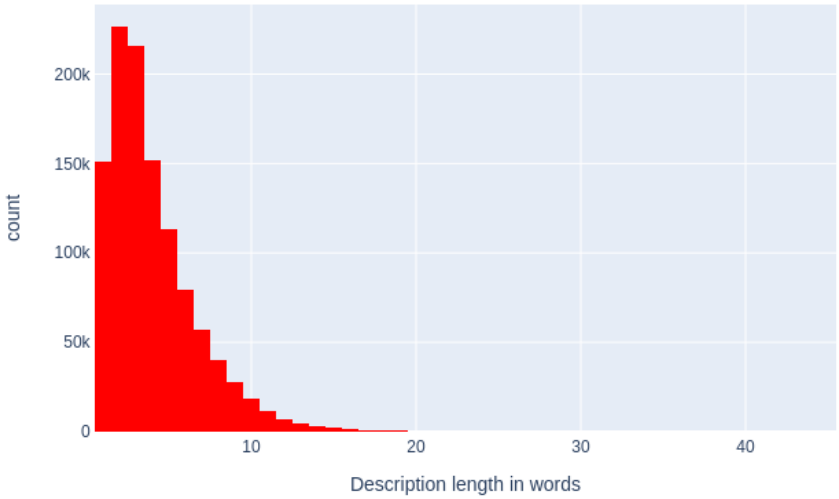

Figure 8. The word frequency distribution histogram after the cleansing

3.1.6 Splitting

Data splitting is a partitioning operation that splits the dataset into training and test chunks used for model learning and evaluation. In multi-class problems, correct splitting aims two purposes: to separate data proportionally and keep the same number of unique classes in both sets.

Stratification: Proportional splitting is achieved by stratified sampling, a method that remains the equal data distribution in test and training sets. To apply stratification, we had to remove additional 876 classes, as it requires each class to have at least two occurrences. Hence, despite a slight change in the number of records, our data only remained 3243 categories. To preserve all classes unique occurrences in both sets, we had to split the data in the proportion of 0.69/0.31, or 776,163 / 348,711 in entries. Figure 9 shows the train and test data distribution plot after the stratification (with the same number of bins):

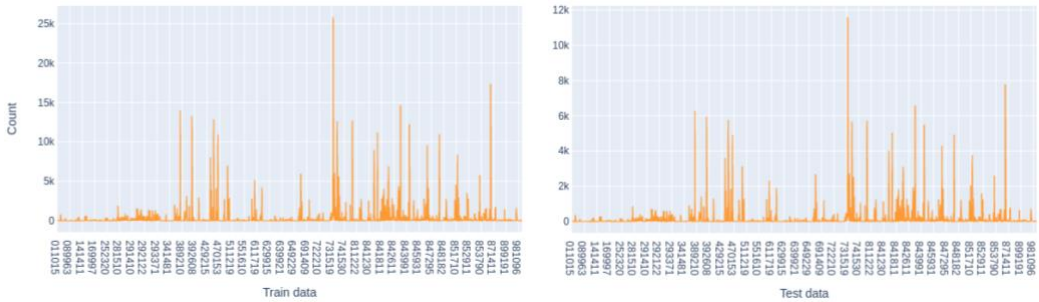

Figure 9. The train and test data classes distribution plots

Rebalancing: Another useful technique frequently used in splitting is data rebalancing. However, we do not consider rebalancing our classes to save its original nature and the models applicability in a real-world context.

3.2 Feature extraction

In text classification systems, feature extraction is a process that transforms pre-cleaned textual tokens into structured feature vectors suitable for the classifier input. In other words, tokenized text sequences must be converted into numerical values to be accepted by the classification model.

There are two primary techniques for determining token weights: weighted words and word embeddings [9]. In this chapter, we review and implement methods from both strategies as well as discuss their advantages and limitations.

3.2.1 Weighted words

Weighted words are a set of techniques for extracting features that assign every token a weight according to specific criteria and transform lists of tokens to vectors of a corpus length. The concept of weighted words is closely related to the Bag-of-Words (BoW) model. While researching this topic, we encountered confusion about its definition, as it is usually mentioned in the context of TF and TF-IDF. To clarify, we consider the BoW nothing but a procedure for extracting bags (lists) of words from the text. Later, they are converted into Bag-of-Features by applying binary, TF, TF-IDF, raw counts, log normalization, or other weighting schemes.

In the upcoming section, we review the two most popular TF and TF-IDF feature extraction schemes as well as discuss their application in our research.

3.2.2 TF and TF-IDF

In our work, TF and TF-IDF value-based feature extraction is the primary technique for simple models such as Decision trees or MLR, and one deep learning model - DNN.

TF is the most basic and straightforward weighting scheme for extracting features from the text. It is computed by counting the number of word's occurrences in each of the text sequences. TF is a robust and cost-effective method, useful for extracting misspelled or unknown words. We do not use it directly in our work, but to calculate the TF-IDF, introduced in the following section. The mathematical representation of TF weight in a text document is formulated in Equation (1):

$$tf(t,d) = f_{t,d} \tag{1}$$

TF-IDF is a TF-based weight factor that has several advantages over TF. TF-IDF decreases the weight assigned to common words, and conversely, increases it for rare ones. According to the study [48], TF-IDF applied in 83% digital libraries of text recommendation systems. The mathematical representation of TD-IDF weight in a text document given in Equation (2):

$$w(d,t) = tf(d,t) * \log\left(\frac{N}{df(t)}\right) \tag{2}$$

Where N is the number of documents and $df(t)$ is the number of documents containing the term t in the vocabulary.

TF-IDF limitations: Despite simplicity and calculation speed, two possible limitations are usually encountered while using TF and TF-IDF values. First, weighed words only consider word multiplicity, not taking into account the order nor semantic properties, a characteristic that differs them from word embeddings. Ignoring these properties may result in considerably lower accuracy. Secondly, their application is limited by the number of unique words in the vocabulary, as it increases the vectors' length and leads to memory-related issues. The latter problem can be partially resolved by taking advantage of sparse matrices, or arrays, where most of the elements are zeros.

3.2.3 Word embedding

Word embeddings are computed numerical representations of text sequences where words with similar meanings have similar weight. Contrary to the weighted words, word embeddings consider both semantic and syntactic values, by determining the distance between words and capturing the position of a word in a sentence. Indeed, these syntactic and semantic meanings have their limitations since word embeddings cannot recognize polysemy, a situation when a word's meaning depends on the context. However, it is still a significant improvement over the capabilities of the weighted words.

In real-world applications, the most popular word embedding algorithms are Word2Vec, Doc2Vec, GloVe, and FastText. Our research particularly uses Word2Vec, Doc2Vec, and GloVe trained vectors as input to CNN. We make use of Gensim, an open-source library that provides its adaptation of both Word2Vec and Doc2Vec. For the GloVe we use the official stand-alone application, which can be downloaded from the Stanford University site [48]. All implemented word embedding models are trained on sets of data before and after the lemmatization, to verify whether the pre-cleansing step removed too many semantic and syntactic text properties.

The most important evaluation criteria of any feature extractor are undoubtedly the classification results. However, it is useful to analyze model output by constructing a two-dimensional Principal Component Analysis (PCA) model of resulting word vectors and visualizing it on the plot. This visual representation allows us to observe the similarities between particular words, evaluate how accurate they were given weights, and compare the layout of different models.

Hence, in the next sections, we briefly introduce all used algorithms, implement them for our datasets, and compare their PCA projections.

3.2.4 Pre-trained word vectors

Conceptionally, listed above algorithms are quite similar and differ only in the implementation details. Before jumping into these details, we need to decide whether we will train model ourselves or use pre-trained word vectors by one of the algorithms on Wikipedia, Google, Twitter, Common Crawl, or other data source.

The words used in our dataset are very domain-specific, and our attempts to use pre-trained models such as Wikipedia trained by Glove were very unsuccessful. The use of pre-made models severely slowed down the training and did not lead to improved accuracy. We believe that the reason for such low results was that models were not directly related to logistics and its jargon's peculiarity. However, this may be due to the ambiguity of HS-6, since the work [54] that studied HS-2 and HS-4 achieved notable results by using pre-trained models.

Thus, due to the poor performance of pre-trained vectors, our further explanation only concerns models trained on our specific dataset.

3.2.5 Word2Vec

Word2Vec is one of the most successful implementations of word embeddings, proposed and developed by Mikolov and researchers from Google [55]. It is a two-layer neural network that utilizes two model architectures: Continuous Bag-of-Words (CBOW) and Continuous Skip-gram. The CBOW makes a weight prediction based on the word's context, and Skip-gram is vice versa, predicts weights of adjacent words based on the current word (Figure 10).

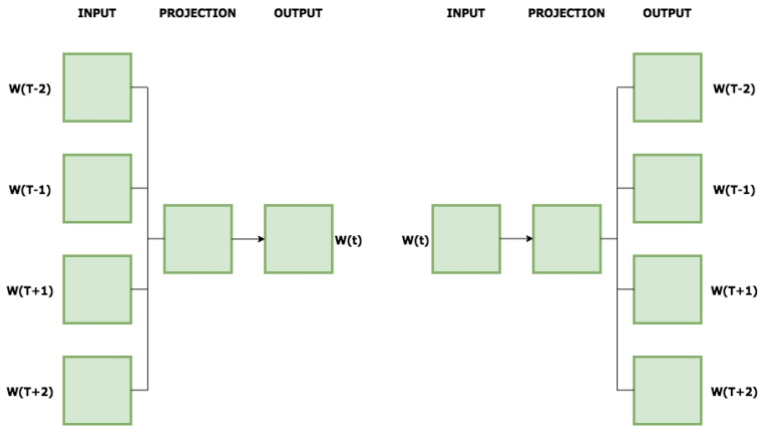

Figure 10. CBOW (left) and Skip-gram (right) model architectures

The list of parameters we used for both Word2Vec implementations in Gensim: {vector_size = 100, iterations = 20, min_count = 1, window=5}. Figure 11 displays the PCA model of the calculated weights by the CBOW method. To avoid text overlapping, it shows only names of the first hundred words. Nevertheless, it is still visible that "acid" and "acetate" are the close neighbors on the plot, indicating that the algorithm correctly determined their semantic meaning.

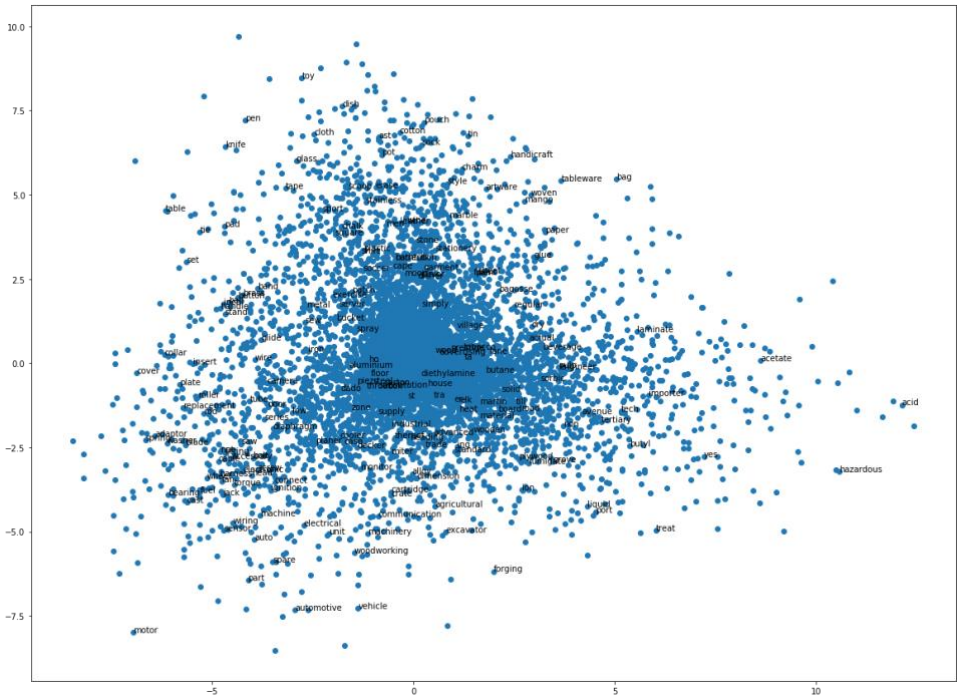

Figure 11. The PCA model of the Word2Vec CBOW trained vectors

Figure 12 shows the same graph for Skip-gram architecture. It has a significantly different graphics layout where same words “acid” and “acetate” are still close to each other, but others such as “engineer” and “excavator” have opposite meanings from the CBOW model.

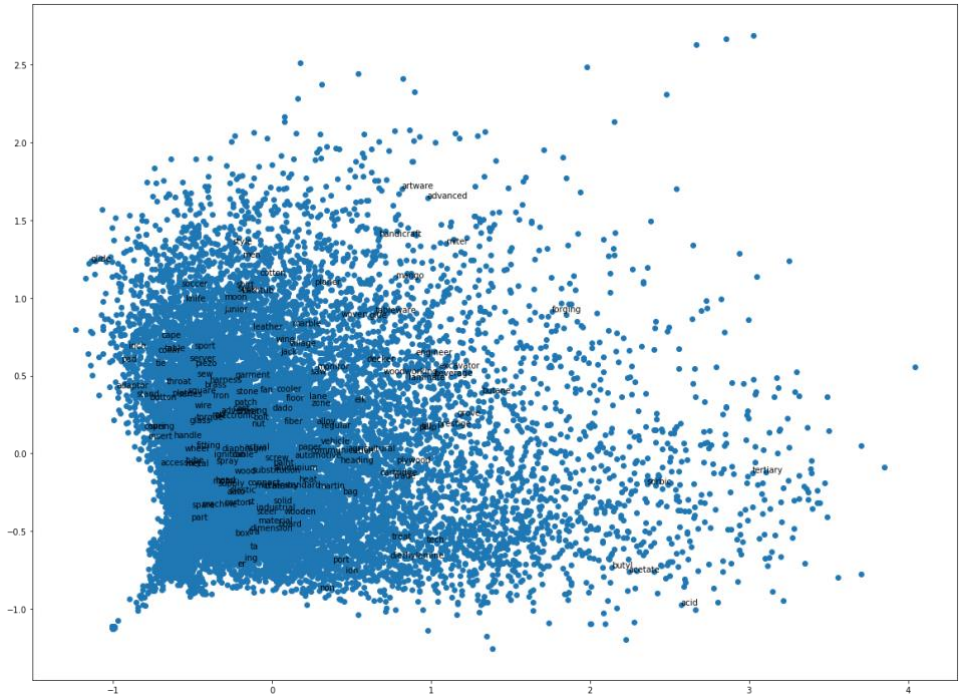

Figure 12. The PCA model of the Word2Vec Skip-gram trained vectors

3.2.6 Doc2Vec

Doc2Vec is an extension to the Word2Vec, built on top of the CBOW and Skip-gram architectures, which are called the Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag-of-Words version of Paragraph Vector (PV-DBOW). The only distinction of these architectures from CBOW and Skip-gram is that after predicting weight, the algorithm concatenates it to a vector of unique paragraph identifiers assigned to every document. Figure 13 shows this difference in the Doc2Vec neural network model scheme, where V is vocabulary size, P is a number of all documents, and N is the size of a hidden layer.

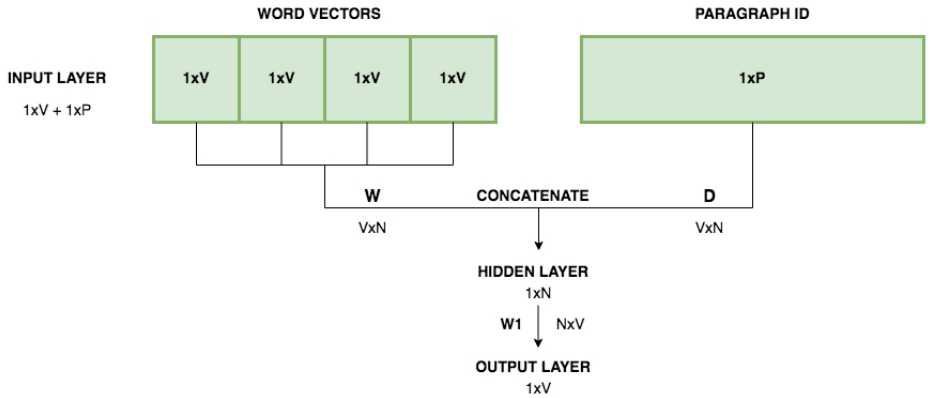

Figure 13. The Doc2Vec network architecture scheme

Thus, Doc2Vec considers the difference in both words and document levels, which, in theory, should better determine the similarities between related product descriptions.

The list of Gensim settings we applied for both Doc2Vec implementations: {vector_size = 100, epochs = 20, min_count = 1, alpha = 0.025, min_alpha = 0.00025}, and manual learning rate decrease by 0.0002 for each epoch. Figure 14 and Figure 15 show the resulting PCA models for both Doc2Vec architectures.

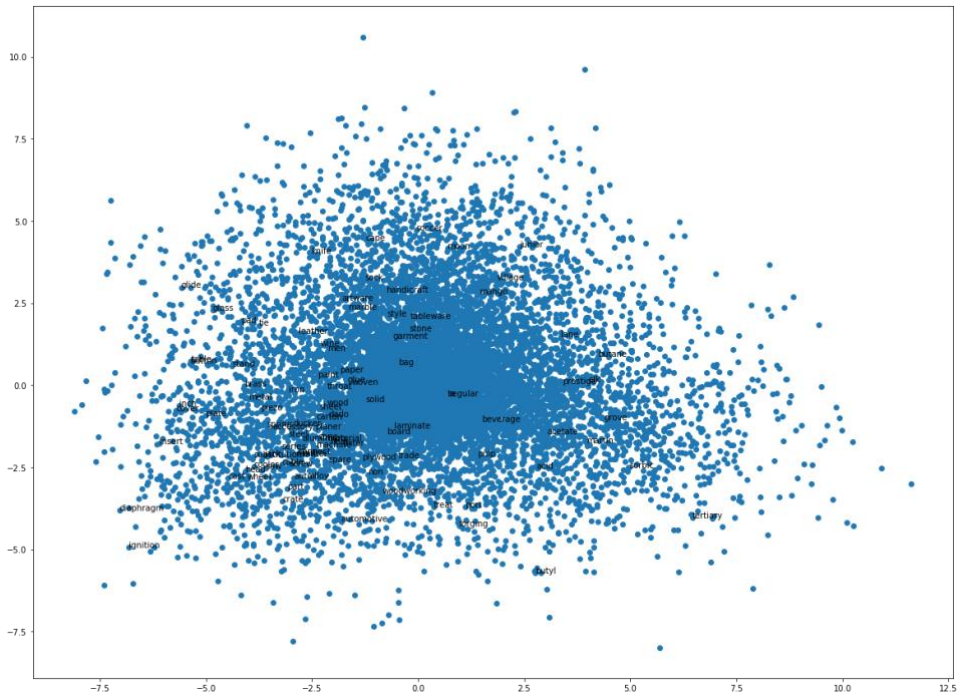

Figure 14. The PCA model of the PV-DM Doc2Vec trained vectors

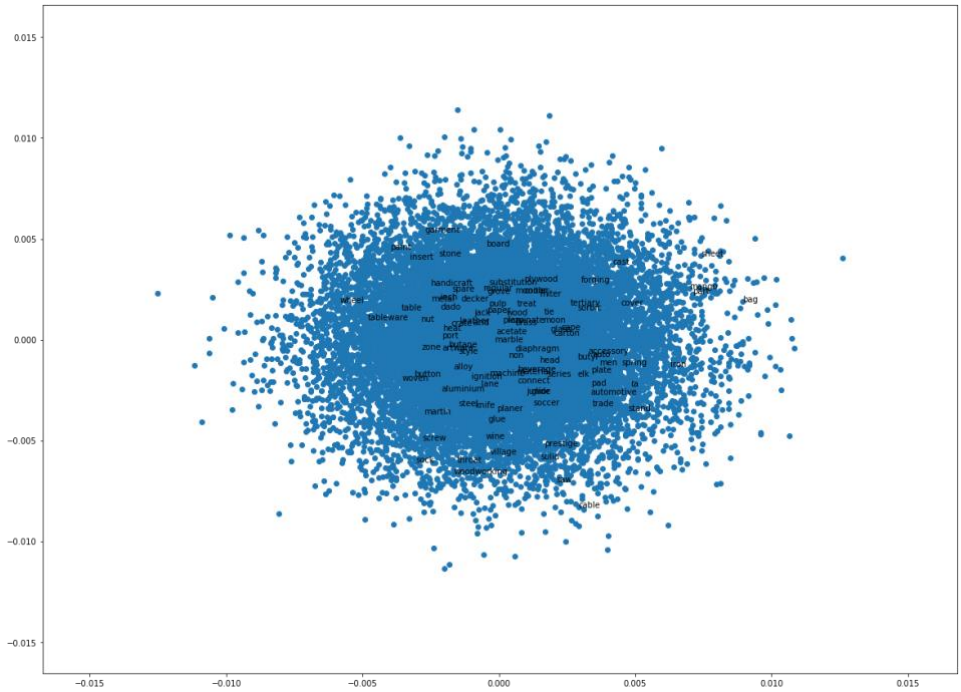

Figure 15. The PCA model of the PV-DBOW Doc2Vec trained vectors

3.2.7 GloVe

Global Vectors for Word Representation (GloVe) is another word embedding technique we use for CNN input. The algorithm is based on a paper [49] published by researchers from Stanford University. Contrary to Word2Vec and Doc2Vec, GloVe is not a predictive neural network, but a count-based model that calculates co-occurrences of words over the whole corpus. In theory, this should work equally well as TF-IDF for our short descriptions, and at the same time, catch the remaining syntactic and semantic meaning.

The list of settings applied in GloVe application: {VOCAB_MIN_COUNT=1, VECTOR_SIZE=100, MAX_ITER=20, WINDOW_SIZE=5, BINARY=2, NUM_THREADS=8, X_MAX=10}. Figure 16 shows the PCA model of the GloVe trained vectors:

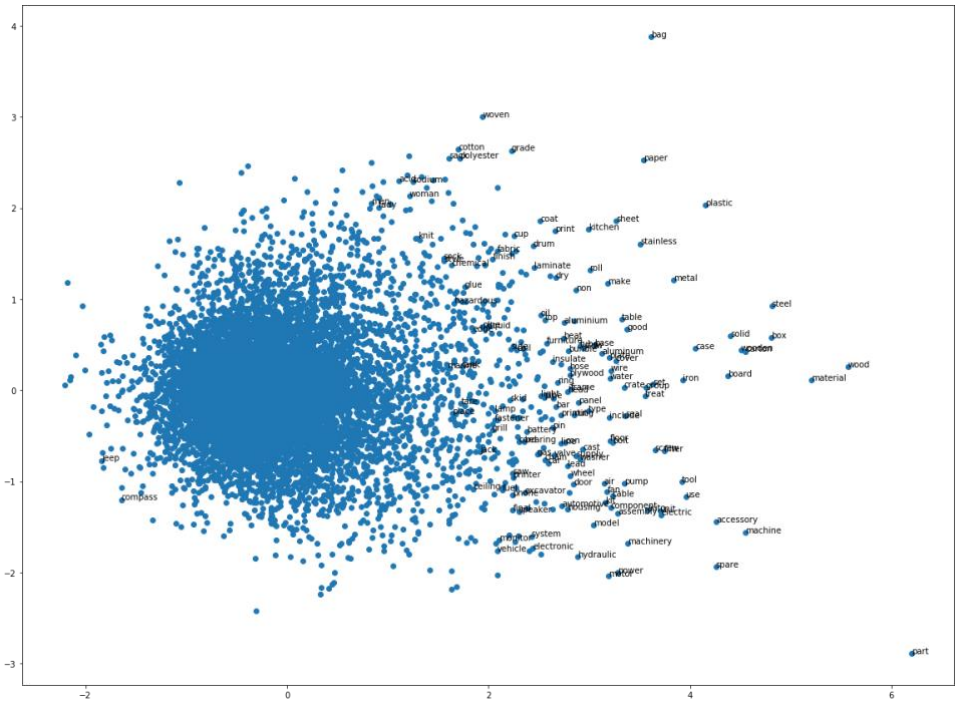

Figure 16. The PCA model of the GloVe trained vectors

3.3 Evaluation

In this chapter, we introduce the metrics and the evaluation strategy applied in our research. These metrics are the leading and the most relevant performance indicators of classification models proposed in our work. Still, before diving into the details, we should understand their limitations and be very cautious with their interpretation for our specific dataset.

As we discovered in the pre-processing data chapter, after the splitting, our dataset contained 3243 unevenly distributed classes and 1,124,874 records. Thus, approximately 8% of initial entries turned into empty lists or were removed at the stratified sampling, and about 35% of remaining records, in essence, became identical. We did not remove these similar records and overall did not rebalance our data to retain natural context and practical relevance. Therefore, all subsequent metrics should be considered in the context of data imbalance and applied text pre-processing.

The metrics we orient when evaluating our models are (1) confusion matrix, (2) precision, (3) recall, (4) F-1 score, (5) support, and their (6) accuracy, (7) weighted and (8) macro averages.

1. **Confusion matrix**: A confusion matrix is a table layout (Figure 17) used to visualize the classification model's performance. The table represents the mapping between negative and positive cases of the predicted and actual classes in the form of the definitions of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). What is essential, this representation is valuable for visualizing both the binary and the multinomial classification performance.

		Predicted class	
		Positive	Negative
Actual class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 17. Confusion Matrix

2. **Precision**: The precision is a classification model metric that shows what percent of all instances classified positive was correct. Mathematically, it is expressed in Equation (3):

$$P = \frac{tp}{tp + fp} \tag{3}$$

3. **Recall**: The recall is a classification model metric that shows what percent of all truly positive instances were classified correctly. The latter can be formulated in Equation (4):

$$R = \frac{tp}{tp + fn} \tag{4}$$

4. **F-1 score**: The F-1 score is one of the primary indicators of the model's performance. It can be described as a harmonic mean of precision and recall, as formulated in Equation (5):

$$F = 2 \times \frac{P \times R}{P + R} \tag{5}$$

5. **Support**: The support shows how many times the class instances occurred in the dataset. This metric stays the same between models and indicates the requirement of rebalancing or stratified sampling.
6. **Accuracy**: Multi-class accuracy is a ratio of correctly predicted instances to the total number of instances, or simply the average of correct predictions. The formula of a multi-class accuracy can be viewed in Equation (6):

$$accuracy = \frac{1}{N}\sum_{k=1}^{|G|} \sum_{x:g(x)=k} I(g(x) = \hat{g}(\mathrm{x})) \tag{6}$$

Here, and in the following formula, *N* is the number of observations, *G* is the set of all classes, and *I* is the characteristic function that returns either one or zero depending on class hit or miss [46]. Despite being the most intuitive and straightforward way to evaluate classification results, accuracy does not consider the imbalance of classes and therefore is not the best performance score for our particular dataset.

7. **Weighted average**: To be equally fair to all individual classes, every class is assigned a weight w_k such that $w_k = \frac{1}{|G|}, \forall k \in \{1, \dots, G\}$, what is described in the Equation (7):

$$weigted\ accuracy = \frac{1}{N}\sum_{k=1}^{|G|} Wi \sum_{x:g(x)=k} I(g(x) = \hat{g}(\mathrm{x})) \tag{7}$$

We consider the weighted average to be the most reliable evaluation score as it counts the proportion of each class.

8. **Macro average**: The macro average does not estimate the proportion of each class in the dataset, but a high macro averaged F-1 score indicates that a classifier adequately performs for every individual class. Equations (8, 9, 10) show the macro-averaged formula of precision, recall, and F-1 score:

$$P_{macro} = \frac{1}{|G|}\sum_{i=1}^{|G|} \frac{tp_i}{tp_i + fp_i} \tag{8}$$

$$R_{macro} = \frac{1}{|G|}\sum_{i=1}^{|G|} \frac{tp_i}{tp_i + fn_i} \tag{9}$$

$$F_{macro} = 2 \times \frac{P_{macro} \times R_{macro}}{P_{macro} + R_{macro}} \tag{10}$$

There are a few other popular metrics, which are frequently used to evaluate the performance of classification models. First of all, it is MCC, ROC, and AUC, mentioned in the background section. However, we found them valuable only for binary classification tasks or when the number of classes was relatively low. The same concerns another standard evaluation score: micro averages of Recall, Precision, and F1-score as they occurred to be misleading for our unbalanced dataset.

Thus, all further classification results are assessed by the final evaluation report matrix (Table 9), with F-1 weighted accuracy and accuracy being the most appropriate performance indicators.

	Precision	Recall	F-1 score	Support
Accuracy			0	0
Macro avg	0	0	0	0
Weighted avg	0	0	0	0

Table 9. Evaluation report matrix

3.4 Classification

Choosing the right classifier is the most critical part of the text classification system. In this chapter, we review some popular algorithms for classifying text and evaluate their performance for predicting HS-6. For clarity, all sections follow the same pattern: we introduce the model's basic theory and our specific implementation details, and then show the evaluation matrix for one or various cases.

Initially, we test non-deep learning classifiers: Rocchio, MLR, MNB, k-NN, Decision Tree, Random Forest, and SVM. By default, all listed algorithms use TF-IDF value-based feature vectors for their model input. Next, we examine deep neural network algorithms: DNN and CNN. CNN is the only model tested on Word2Vec, Doc2Vec, and GloVe extracted feature vectors and studied on two sets of differently cleansed data: lemmatized and without lemmatization. DNN, on the other hand, uses a standard TF-IDF weighting scheme. To find the most performant model, we utilize the Random Multimodel Deep Learning for Classification (RMDL), a tool based on researches [56][57], which we discuss in detail in a separate section.

All experiments were conducted in Jupyter and Spyder using Keras and Scikit-learn open-source libraries, and utilizing Intel Xeon E5-2680 v2, 10 core and 20 thread processor with 2.8-3.6 Hz frequency, 25M L3 cache, and 32GB of RAM.

3.4.1 Rocchio classification

Introduced by J.J. Rocchio [32], Rocchio classification is a method that applies relevance feedback to query full-text information retrieval systems. The algorithm determines class boundaries by building a prototype vector for each class on a training set of documents, assigning each test document a class by defining the most similar prototype vector, and measuring the distance between calculated centroids (centers of mass of their members) for every class.

Despite the algorithm's universality and computational speed (almost instant), Rocchio has issues with classifying multi-class data, confirmed by our results. Rocchio is the only algorithm tested in our research where weighted F-1 average overcame the standard accuracy.

	Precision	Recall	F-1 score	Support
Accuracy			0.26	348711
Macro avg	0.18	0.28	0.16	348711
Weighted avg	0.58	0.26	0.31	348711

Table 10. Rocchio classifier evaluation report

3.4.2 Multinomial Logistic Regression

Multinomial Logistic Regression (MLR) classification algorithm generalizes traditional binomial Logistic Regression (LR) to multi-class applications. MLR does not predict classes but rather probabilities of various possible outcomes of categorically distributed dependent variable, given one or multiple independent variables [34].

MLR classification did not have memory problems but took approximately 12 hours to calculate, proving to be adequate for multi-class tasks.

	Precision	Recall	F-1 score	Support
Accuracy			0.56	348711
Macro avg	0.25	0.16	0.18	348711
Weighted avg	0.57	0.56	0.54	348711

Table 11. MLR classifier evaluation report

3.4.3 Multinomial Naïve Bayes

In general, Naïve Bayes classifier is a Bayes' theorem-based method that assumes naïve or strong conditional independence between the features [35]. Multinomial Naïve Bayes classifier is a particular case of Naïve Bayes that uses a multinomial distribution for each feature.

MNB occurred to be a very memory-consumable model that also delivered poor results. To fit our data into MNB, we had to apply partial fit and transform TF-IDF sparse matrices every iteration.

	Precision	Recall	F-1 score	Support
Accuracy			0.43	348711
Macro avg	0.11	0.04	0.05	348711
Weighted avg	0.50	0.43	0.38	348711

Table 12. MNB classifier evaluation report

3.4.4 K-Nearest Neighbor

K-Nearest Neighbors (k-NN) is a non-parametric algorithm that attempts to predict a class by searching for the closest or k-defined data points [36]. Thus, behind the k-NN lies the simple idea that related data points are most often located close to each other. The performance of k-NN may indirectly reflect how well was conducted data pre-processing, as it relies on measuring the distance between nearby features.

K-NN turned out to be the fastest classification algorithm in our research, also achieving great results for such a simple model. Besides, these results indirectly proved the effectiveness of our pre-cleansing step.

	Precision	Recall	F-1 score	Support
Accuracy			0.59	348711
Macro avg	0.31	0.28	0.27	348711
Weighted avg	0.60	0.59	0.59	348711

Table 13. k-NN classifier evaluation report

3.4.5 Decision tree

The decision tree classification model predicts a class by utilizing a tree structure that represents leaves as categorical values and branches as a set of associated features led to those values [40]. Such tree composition is achieved through binary recursive partitioning, a process that iteratively breaks the data into partitions based on feature-dependent splitting rules. Decision tree classifier needs little data pre-processing and usually works well with large datasets.

In our experiments, the decision tree model did not struggle with overfitting and proved to be a very robust and effective way of classifying HS-6 codes.

	Precision	Recall	F-1 score	Support
Accuracy			0.60	348711
Macro avg	0.34	0.29	0.30	348711
Weighted avg	0.60	0.60	0.59	348711

Table 14. Decision tree classifier evaluation report

3.4.6 Random forest

Random forest classifier is an ensemble learning algorithm that makes a category prediction by building numerous decision trees and calculating the classes mode for each built tree [41]. Random forests reduce the problem of training data overfitting frequently encountered by decision trees, what in theory, should lead to better results. However, random forests generally have issues with large datasets as they grow too fast in memory.

As a result, Random forests required too much memory, and we could not find a proper way to train them incrementally: warm start method always encountered issues, despite incrementing the number of estimators each iteration. The maximum tree depth we managed to achieve without kernel crashes were the following parameters: {n_estimators=100, max_depth=35}. Unfortunately, such an insufficient depth was not enough for adequate results, which is a matter of further exploration with more RAM, or correctly configured warm starts.

	Precision	Recall	F-1 score	Support
Accuracy			0.35	348711
Macro avg	0.17	0.04	0.05	348711
Weighted avg	0.61	0.35	0.34	348711

Table 15. Random forest classifier evaluation report

3.4.7 Support Vector Machine

Invented by Vapnik and Chervonenkis in 1963, Support Vector Machine (SVM) is a classification algorithm that defines boundaries between classes by utilizing a set of hyperplanes in infinite-dimensional space [38]. SVM is especially powerful in text categorization as it does not require too many labeled training instances, which might be helpful for our unbalanced dataset.

SVM model took a few hours to train and did not encounter any memory-related problems, demonstrating considerably high results.

	Precision	Recall	F-1 score	Support
Accuracy			0.58	348711
Macro avg	0.35	0.27	0.28	348711
Weighted avg	0.58	0.58	0.56	348711

Table 16. SVM classifier's evaluation report

3.4.8 RMDL

RDML is a technique used to search for the most accurate CNN, RNN, and DNN deep learning classification models, proposed by K. Kowsari, M. Heidarysafa, and other researchers [55][56]. RDML architecture consists of CNN, DNN, and RNN layers (Figure 18), that generate models with an arbitrary number of hidden layers, nodes, and the optimizer type. Each randomly generated model is then trained and compared with the best previously found structure until it discovers the most accurate classification model. This iterative way allows deep learning models to achieve state-of-art results for various text and image classification problems.

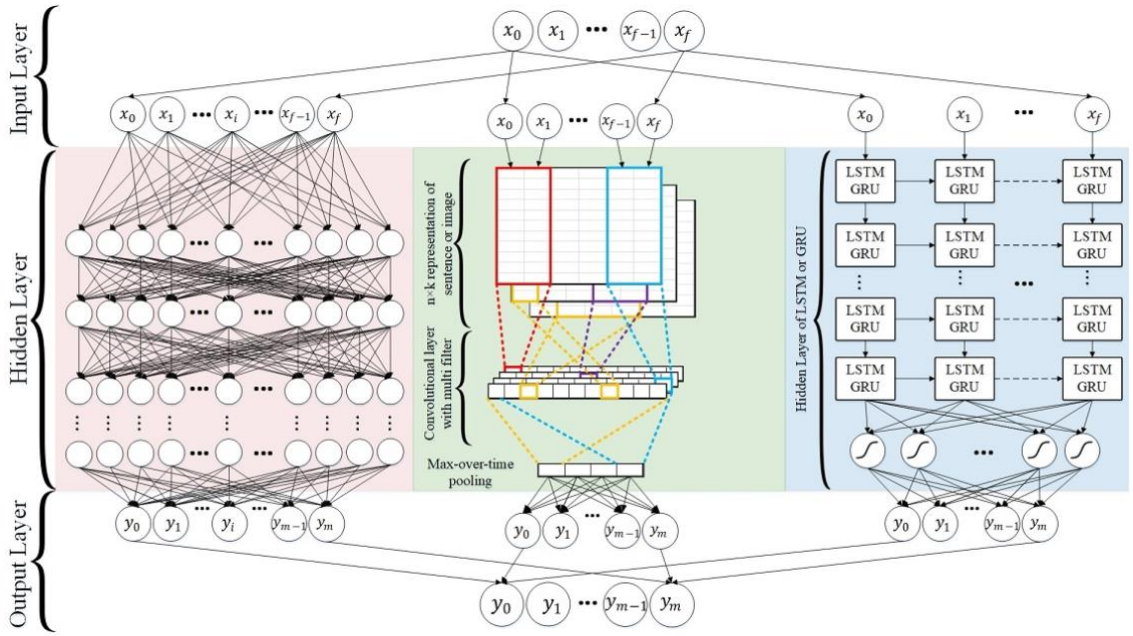

Figure 18. RDML architecture

In order to utilize the RDML approach for our specific task, we added (1) Word2Vec and Doc2Vec support, solved (2) memory-related issues, and (3) trained DNN and CNN models for some time to gain an intuition about suitable architecture parameters.

1. **Word2Vec and Doc2Vec support**: First, we added the possibility of Word2Vec and Doc2Vec CNN model input as RDML initially only worked with TF-IDF and GloVe vectors.
2. **Memory-related issues**: RDML was not designed to work with relatively large datasets, and our first attempts to train models resulted in crashes. Kernel crashes primarily concerned DNN that required a lot of RAM both for the model and TF-IDF based vectors input. To solve the issue, we decided to train our models in small batches by unwrapping sparse matrices inside Keras fit generators. In theory, this approach does not impact the performance, as small batches tend to achieve the same accuracy as large ones and even show less model degradation [58] (Figure 19).

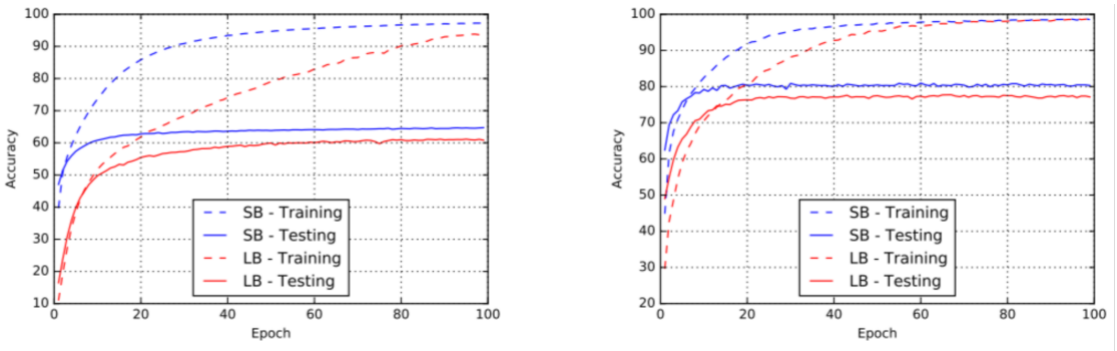

Figure 19. Two samples of convergence trajectories of training and testing accuracy for small (SB) and large batches (LB).

In addition, after every epoch, we saved the model and cleaned the memory by the garbage collector. As a result, we were able to train models with our limited memory resources.

3. **DNN and CNN training**: Both models took approximately a day to learn until they started overfitting. Due to these time constraints, we did not achieve the best possible results directly in RDML but rather gained an intuition about the models settings. We observed that for DNN two hidden layers, 512-1024 nodes, and "RMSprop" optimizer give better outcomes. As for CNN, it was five hidden layers, 128 nodes, and "adam" optimizer. Therefore, further training was done manually within found boundaries.

RDML observations could be misleading, and we do not consider found models to be the state-of-art results, requiring more exploration. Better results could be achieved by faster training utilizing GPU, which in our case is very poor, dimensionality reduction, or more learning time. Thus, the resulting models reviewed in subsequent sections are just a starting point for improvements and further research.

3.4.9 DNN

A deep neural network is an artificial neural network that calculates the probability of each outcome by passing through multiple hidden layers [42]. Each hidden layer generates a map of virtual neurons with arbitrary weights between them, that are then adjusted each epoch. The number of hidden layers and neurons is determined by the data and type of features vectors. In our case, their amount was discovered using the RDML and general intuition. Figure 20 shows the summary of DNN that achieved the best results during our experiments:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 839)	14,653,135
dropout (Dropout)	(None, 839)	0
dense_1 (Dense)	(None, 839)	704,760
dropout_1 (Dropout)	(None, 839)	0
dense_2 (Dense)	(None, 839)	704,760
dropout_2 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 3243)	2,724,129

Total params: 39,461,396
Trainable params: 39,461,396
Non-trainable params: 0

Figure 20 DNN model summary

	Precision	Recall	F-1 score	Support
Accuracy			0.62	348711
Macro avg	0.31	0.23	0.25	348711
Weighted avg	0.63	0.62	0.61	348711

Table 17. DNN classifier evaluation report

3.4.10 CNN

A convolutional neural network is a deep neural network whose neurons in all layers are fully connected to each other [42]. The layers relatedness makes CNNs less vulnerable to overfitting data, frequently struggled by DNNs. Also, with CNNs, on the contrary to DNNs, we did not face any memory issues as it used vectors extracted by word embedding algorithms.

We trained CNNs on Word2Vec CBOW and Skip-gram, Doc2Vec PV-DM and PV-DBOW, and GloVe models on two sets of data: with and without lemmatization. Without lemmatization, it showed very weak performance (maximum 35% accuracy), omitted in the final evaluation reports. Thus, all results reviewed in this section were accomplished by using lemmatized data. Figure 21 shows the summary of CNN that achieved the best performance during our experiments:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 22)]	0	
embedding (Embedding)	(None, 22, 100)	1,746,500	input_1[0][0]
reshape (Reshape)	(None, 22, 10, 10)	0	embedding[0][0]
conv2d (Conv2D)	(None, 22, 10, 128)	5248	reshape[0][0]
conv2d_1 (Conv2D)	(None, 22, 10, 128)	11,648	reshape[0][0]
conv2d_2 (Conv2D)	(None, 22, 10, 128)	20,608	reshape[0][0]
conv2d_3 (Conv2D)	(None, 22, 10, 128)	32,128	reshape[0][0]
conv2d_4 (Conv2D)	(None, 22, 10, 128)	46,208	reshape[0][0]
average_pooling2d (AveragePooling2D)	(None, 5, 10, 128)	0	conv2d[0][0]
average_pooling2d _1 (AveragePooling2D)	(None, 5, 10, 128)	0	conv2d_1[0][0]
average_pooling2d _2 (AveragePooling2D)	(None, 5, 10, 128)	0	conv2d_2[0][0]
average_pooling2d _3 (AveragePooling2D)	(None, 5, 10, 128)	0	conv2d_3[0][0]
average_pooling2d _4 (AveragePooling2D)	(None, 5, 10, 128)	0	conv2d_4[0][0]
concatenate (Concatenate)	(None, 25, 10, 128)	0	average_pooling2d[0][0] average_pooling2d_1[0][0] average_pooling2d_2[0][0] average_pooling2d_3[0][0] average_pooling2d_4[0][0]
conv2d_5 (Conv2D)	(None, 25, 10, 128)	409,728	concatenate[0][0]
average_pooling2d _5 (AveragePooling2D)	(None, 5, 5, 128)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 5, 5, 128)	409,728	average_pooling2d_5[0][0]
average_pooling2d _6 (AveragePooling2D)	(None, 1, 3, 128)	0	conv2d_6[0][0]
dropout (Dropout)	(None, 1, 3, 128)	0	average_pooling2d_6[0][0]
flatten (Flatten)	(None, 384)	0	dropout[0][0]
dense (Dense)	(None, 128)	49,280	flatten[0][0]
dropout_1 (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 3243)	418,347	dropout_1[0][0]

Total params: 3,149,423
Trainable params: 3,149,423
Non-trainable params: 0

Figure 21. CNN model summary

	Precision	Recall	F-1 score	Support
Accuracy			0.56	348711
Macro avg	0.20	0.14	0.15	348711
Weighted avg	0.60	0.56	0.54	348711

Table 18. Word2Vec CBOW evaluation report

	Precision	Recall	F-1 score	Support
Accuracy			0.57	348711
Macro avg	0.20	0.14	0.15	348711
Weighted avg	0.60	0.57	0.55	348711

Table 19. Word2Vec Skip-gram evaluation report

	Precision	Recall	F-1 score	Support
Accuracy			0.56	348711
Macro avg	0.19	0.13	0.14	348711
Weighted avg	0.59	0.56	0.53	348711

Table 20. Doc2Vec PV-DM evaluation report

	Precision	Recall	F-1 score	Support
Accuracy			0.56	348711
Macro avg	0.20	0.14	0.15	348711
Weighted avg	0.59	0.56	0.54	348711

Table 21. Doc2Vec PV-DBOW evaluation report

	Precision	Recall	F-1 score	Support
Accuracy			0.57	348711
Macro avg	0.21	0.14	0.15	348711
Weighted avg	0.59	0.57	0.55	348711

Table 22. GloVe evaluation report

4 Results

After done preparation, cleansing, and stratified random splitting, our data remained 1,124,874 out of 1,220,834 entries. Most of these records were dropped either because they did not contain any product-related keywords or required more than a single class occurrence at the splitting. After the removal, the data retained 3243 of the initial 4328 unequally distributed HS-6 categories, which were not rebalanced to keep the original context.

Here, we present the resulting table for different combinations of classifiers and feature extractors examined in our research. Their performance is assessed and sorted by an F-1 weighted average metric, a score that considers every class proportionality.

Classifier	Features	F-1 macro avg	F-1 accuracy	F-1 weighted avg
DNN	**TF-IDF**	**0.25**	**0.62**	**0.61**
Decision tree	TF-IDF	0.30	0.60	0.59
k-NN	TF-IDF	0.27	0.59	0.59
SVM	TF-IDF	0.28	0.58	0.56
CNN	GloVe	0.15	0.57	0.55
CNN	Word2Vec Skip-gram	0.15	0.57	0.55
MLR	TF-IDF	0.18	0.56	0.54
CNN	Word2Vec CBOW	0.15	0.56	0.54
CNN	Doc2Vec PV-DBOW	0.15	0.56	0.54
CNN	Doc2Vec PV-DM	0.14	0.56	0.53
MNB	TF-IDF	0.05	0.43	0.38
Random Forests	TF-IDF	0.05	0.35	0.34
Rocchio	TF-IDF	0.16	0.26	0.31

Table 23. The resulting evaluation matrix

5 Discussion

5.1 Dataset analysis

Before analyzing the results, we want to discuss the extreme noisiness of the dataset investigated in our thesis. The pre-processing step showed that the average number of words and symbols after the cleansing decreased by 76% and 74%, and the number of overall words reduced from 20,265,061 to 5,158,174. The received statistics mean that only about 25% of descriptions length had keywords connected to product definitions. In addition, 8% or 95,960 entries did not contain any product-related words at all.

The variety of words in descriptions also drastically dropped: the final vocabulary contained only 17,476 words contrary to 1,405,333 words before the cleansing procedures, or approximately 1.24% of all original unique words. Most of the word diversity was expelled by the lemmatization and the removal of non-English words, which unified words with the same roots, and excluded typos, abbreviations, and other unnecessary keywords. As a result, the received dictionary describes every product class on average in five words.

Thus, the remaining descriptions held on the average 4 words or 25 symbols with a variety of 17,476 unique words. Such a significant decrease in overall and unique words means that the final descriptions applied in classification were too short and not very representative. The latter is very sensitive for HS-6 level codes studied in our thesis, as they usually have minor differences between the neighboring product definitions. This tendency is visible at the distribution plot of our 3243 classes, where most codes are namely concentrated in specific bordering areas.

The above analysis does not consider the fact that most product codes in the harmonized number column are originally misclassified; we take their correctness for granted. However, as we stated in the introduction, the classification done by booking officers and sellers is most often incorrect, and especially it concerns HS-6 codes with its similarities between adjacent descriptions, sometimes hard to distinguish even having physical goods.

5.2 Classification analysis

This section discusses and compares the obtained classification results in terms of preparation, feature extraction, implementation complexity, and real-world applicability. In the end, we try to validate and interpret the received results to real-world applicability.

5.2.1 Lemmatized vs non-lemmatized

At the pre-processing step, we were doubtful about using data lemmatization for word embeddings, as we did not want to lose important semantic and syntactic meanings. For this purpose, we maintained two sets of data, expecting better results in further CNN training. However, CNN, trained on features vectors extracted by the Word2Vec algorithm from non-lemmatized data, started overfitting at about 35% accuracy. Therefore, considering other algorithms similarities and long training processes of both the CNN and word embedding models themselves, we decided to stop further explorations and ignored these results in the final table.

Low results without the lemmatization are explained by the quality of our data. As we asserted in the dataset analysis, most descriptions are too short and simply do not hold any semantic or syntactic properties. As a result, a lemmatization in combination with POS-tagging and the removal of basic English forms gives much better performance outcomes for both weighted words and word embeddings.

5.2.2 Weighted words vs Word embeddings

In our research, we used weighted words for all the models except CNN, which applied five variations of word embedding algorithms. Overall, TF-IDF weights delivered equal and even slightly better accuracy than word embeddings as our short descriptions contained too little context for them to be efficient. At the same time, this difference is not very significant and, in theory, could be compensated by finding a better CNN model.

From the implementation perspective, both approaches had certain difficulties. Word embeddings did not have problems fitting into memory as their dimensions were much smaller than TF-IDF matrices, but were a little harder to implement to be input to the CNN. Weighted words are vice versa, had huge issues with memory, but were relatively easy to put into classification models. In order to solve memory-related problems for some models such as MNB and DNN, we had to struggle with unzipping sparse matrices in fit generators, and training models in very small batches. In case it is not important to utilize the original dataset, another way to solve these problems could be using dimensionality reduction techniques.

5.2.3 Doc2Vec vs Word2Vec vs GloVe

Our CNN model was taught on three various word embedding algorithms and their specific architectures: Doc2Vec PV-DM and PV-DBOW, Word2Vec CBOW and Skip-gram, and GloVe. The classification results did not show any significant differences between these models, except that Word2Vec Skipgram and GloVe displayed slightly better accuracy, weighted average, and macro average. However, this difference could simply be a matter of training deviations and should not be considered serious.

5.2.4 Linear vs deep learning models

The classification results showed that the DNN classifier with TF-IDF extracted features had the highest F-1 weighted average score of 61% and 62% accuracy. CNNs with word embeddings, on the other hand, were slightly behind Decision tree, k-NN, SVM, and MLR classifiers, achieving only a 55% weighted average and 57% accuracy by GloVe. Thus, most linear models displayed approximately the same weighted accuracy as deep learning models, except Random Forests, MNB, and Rocchio classifiers. However, despite high results, linear models reached their limits and cannot be further progressed on the contrary to deep neural networks, which are subject to improvements by RDML.

In terms of implementation complexity, linear models almost did not require any additional configuration and were considerably fast to compute. The only exceptions were MNB and Random Forests, which after the model adjustment and partial fit, occurred not very effective for classifying so many classes. At the same time, DNNs and CNNs took an enormous time to implement and to train, and we still did not manage to find the best models.

5.2.5 Validation

The validation of the received results is the subject to practice and the applicability in real-world EU customs system. However, due to the lack of such an opportunity, we attempt to validate these results according to the existing data. As we stated in the introduction, the volume of tax underpayment related to HS code misclassification in the EU is roughly estimated at 65% of all imported commodities [4]; hence, the accuracy of current customs methods in detecting real price is about 35%. Our study covers a significant part of this nomenclature (3243 classes), and assuming there is a direct link between the wrong code and the price, that number should increase to 61%. Of course, this is a very naive conclusion that has nothing to do with reality, but with utilizing proposed ML algorithms, it would at least be possible to assign weights to unreliable companies or find another useful way to detect fraud. Moreover, these models can be further improved by applying less noisy data or using RMDL, which we did not fully utilize due to the time and hardware constraints. Thus, despite the dataset quality and relatively low 61% weighted accuracy, the received results are a solid starting point for using ML algorithms in the actual EU customs HS-6 classification systems or the currently developed OSS.

6 Summary

The aim of this master's thesis was to examine the efficiency of machine learning algorithms for classifying HS-6 codes according to their cargo descriptions. To achieve this goal, we utilized the US Import 2018-2020 dataset provided by Enigma that initially held about 41,000,000 records. After the removal of duplicates, corrupted values, HS-2, HS-4, and HTS codes, the dataset only remained nearly 1,200,000 entries. This data was then applied to the cleansing, which included tokenization and a list of normalization procedures: converting to lower case, POS tagging, lemmatization, and removing non-English words, punctuation, stop words, domain-related noise, textual digits, non-ASCII, and non-alpha characters. Finally, after the stratified sampling and splitting, the dataset kept 3243 product classes and the vocabulary of 17,476 unique words in the descriptions. The remaining records were extracted feature weights using weighted words and word embedding algorithms: TF-IDF and Doc2Vec, Word2Vec, and GloVe. The received feature vectors were used as input to classification models: Rocchio, MLR, MNB, k-NN, Decision tree, Random forest, SVM, DNN, and CNN classifiers. DNN and CNN deep neural networks were trained using the RMDL approach that enabled us to find the most performant classification model in our research. As a result, the highest 61% F-1 weighted average and 62% accuracy were achieved by the DNN model.

Thus, we classified 3243 distinct HS-6 classes according to 1,124,874 cargo descriptions using Rocchio, MLR, MNB, k-NN, Decision tree, Random forest, SVM, and DNN, and CNN classifiers with TF-IDF, Word2Vec, Doc2Vec, and GloVe extracted feature vectors and achieved 61% F-1 weighted accuracy by the DNN model. Considering the extreme noisiness of the dataset, shortness and ambiguity of HS-6 descriptions, and the overall amount of classes, 61% proportional accuracy is a significant achievement. However, this could still be further improved by utilizing RMDL and speeding up computations by dimensionality reduction or GPU training.

All the relevant materials, Jupyter notebooks with all models outputs and code for each chapter are available at https://github.com/denissruder/master_thesis.

7 Bibliography

[1] "Ecommerce in Europe: €621 billion in 2019," 2019. https://ecommercenews.eu/ecommerce-in-europe/ (accessed Jun. 24, 2020).

[2] "Year-on-year growth in weekly online orders in the retail industry during the Coronavirus pandemic in selected countries in Europe in 2020," 2020. https://www.statista.com/statistics/1109296/online-retail-y-o-y-order-trends-during-coronavirus-in-europe (accessed Jun. 24, 2020).

[3] "Collection of VAT and customs duties on cross-border e-commerce," 2018. https://www.eca.europa.eu/Lists/ECADocuments/BP_VAT/BP_VAT_EN.pdf (accessed Jun. 24, 2020).

[4] Basalisco Bruno, Wahl Julia, and Okholm Henrik, "E-commerce imports into Europe: VAT and customs treatment," *Copenhagen Economics*, 2016.

[5] European Court of Auditors, "E-commerce: many of the challenges of collecting VAT and customs duties remain to be resolved," 2019. https://www.eca.europa.eu/lists/ecadocuments/sr19_12/sr_e-commerce_vulnerability_to_tax_fraud_en.pdf (accessed Jun. 24, 2020).

[6] European Commission, "Regulation (EU) 2020/194 of 12 February 2020," 2020. https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1582118248440&uri=CELEX%3A32020R0194 (accessed Jun. 24, 2020).

[7] European Taxation and Customs Union, "Modernising VAT for cross-border e-commerce," 2020. https://ec.europa.eu/taxation_customs/business/vat/modernising-vat-cross-border-ecommerce_en (accessed Jun. 24, 2020).

[8] World Customs Organization, "The Harmonized Commodity Description and Coding System." http://www.wcoomd.org/en/topics/nomenclature/overview/what-is-the-harmonized-system.aspx (accessed Jun. 24, 2020).

[9] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, "Text Classification Algorithms: A Survey," *Information (Switzerland)*, vol. 10, no. 4, Apr. 2019.

[10] C. C. Aggarwal and C. X. Zhai, "A survey of text classification algorithms," in *Mining Text Data*, vol. 9781461432234, Springer US, pp. 163–222, 2012.

[11] G. Gupta and S. Malhotra, "Text Document Tokenization for Word Frequency Count using Rapid Miner (Taking Resume as an Example)." *IJCA Proceedings on International Conference on Advancements in Engineering and Technology*, 2015.

[12] T. Verma, R. Renu, and D. Gaur, "Tokenization and Filtering Process in RapidMiner," *International Journal of Applied Information Systems*, vol. 7, no. 2, pp. 16–18, Apr. 2014.

[13] V. Gupta and G. S. Lehal Professor, "A Survey of Text Mining Techniques and Applications," *Journal of Emerging Technologies in Web Intelligence*, vol. 1, no. 1, pp. 60-76, Aug. 2009.

[14] M. K. Dalal and M. A. Zaveri, "Automatic Text Classification: A Technical Review," 2011.

[15] B. Pahwa, S. Taruna, and N. Kasliwal, "Sentiment Analysis- Strategy for Text Pre-Processing," *International Journal of Computer Applications*, vol. 180, no. 34, pp. 15–18. 2018.

[16] J. Singh and V. Gupta, “Text stemming: Approaches, applications, and challenges,” *ACM Computing Surveys*, vol. 49, no. 3, pp. 1–46, Sep. 2016.
[17] J. Dziadek, A. Henriksson, and M. Duneld, “Improving Terminology Mapping in Clinical Text with Context-Sensitive Spelling Correction.,” *Studies in health technology and informatics*, vol. 235, pp. 241–245, Jan. 2017.
[18] M. V. Christanti, Rudy, and D. S. Naga, “Fast and accurate spelling correction using trie and Damerau-levenshtein distance bigram,” *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 16, no. 2, pp. 827–833, Apr. 2018.
[19] J. Plisson, J. Plisson, N. Lavrac, and D. Mladenic, “A rule based approach to word lemmatization,” *Proceesings of ISO4*, 2004.
[20] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, “Stemming and lemmatization in the clustering of finnish text documents,” *International Conference on Information and Knowledge Management, Proceedings*, pp. 625–633, 2004.
[21] Y. Goldberg and O. Levy, “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method,” *CoRR*, Feb. 2014.
[22] Q. Le and T. Mikolov, “Distributed Representations of Sentences and Documents.” *CoRR*, 2014.
[23] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.
[24] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of Tricks for Efficient Text Classification,” *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, vol. 2, pp. 427–431, Jul. 2016.
[25] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, Jul. 2016.
[26] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, Jan. 1988.
[27] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4. John Wiley & Sons, Ltd, pp. 433–459, Jul. 01, 2010.
[28] S. Balakrishnama and A. Ganapathiraju, “Linear Discriminant Analysis—A Brief Tutorial,” vol. 11, Jun. 1998.
[29] M. Sugiyama, “Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis.,” *Journal of Machine Learning Research*, vol. 8, pp. 1027–1061, Jun. 2007.
[30] F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons, “Document clustering using nonnegative matrix factorization,” *Information Processing & Management*, vol. 42, no. 2, pp. 373–386. Nov. 2004.
[31] S. Tsuge, M. Shishibori, S. Kuroiwa, and K. Kita, “Dimensionality reduction using non-negative matrix factorization for information retrieval,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 960–965, 2001.
[32] J. Rocchio, “Relevance feedback in information retrieval,” 1965.
[33] F. E. Harrell and D. G. Levy, “Regression Modeling Strategies,” 1995.

[34] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression: Third Edition*. Wiley, 2013.
[35] R. R. Larson, "Introduction to Information Retrieval," *Journal of the American Society for Information Science and Technology*, 2009.
[36] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2888, pp. 986–996, 2003.
[37] M. S.-D. and P. J. S. Jasper J. Koehorst, Jesse C. J. van Dam, Edoardo Saccenti, Vitor A. P. Martins dos Santos, "GISAID Global Initiative on Sharing All Influenza Data. Phylogeny of SARS-like betacoronaviruses including novel coronavirus (nCoV)," *Oxford*, vol. 34, no. 8, pp. 1401–1403, 2017.
[38] L. M. Manevitz, M. Yousef, N. Cristianini, J. Shawe-Taylor, and B. Williamson, "One-Class SVMs for Document Classification," 2001.
[39] E. H. S. Han and G. Karypis, "Centroid-based document classification: Analysis and experimental results." Springer- Verlag, pp. 424–431, Jan. 01, 2000.
[40] W. M.U. Noormanshah, P. N.E. Nohuddin, and Z. Zainol, "Document Categorization Using Decision Tree: Preliminary Study," *International Journal of Engineering & Technology*, vol. 7, no. 4.34, p. 437, Dec. 2018.
[41] B. Xu, X. Guo, Y. Ye, and J. Cheng, "An improved random forest classifier for text categorization," *Journal of Computers (Finland)*, vol. 7, no. 12, pp. 2913–2920, 2012.
[42] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks*, vol. 61, pp. 85–117, Apr. 2014.
[43] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading Text in the Wild with Convolutional Neural Networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, Jan. 2016.
[44] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, 2015.
[45] D. P. Mandic and J. A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures andStability*. 2001.
[46] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, Mar. 2005.
[47] C. Goutte and E. Gaussier, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation," in *Lecture Notes in Computer Science*, vol. 3408, pp. 345–359, 2005.
[48] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, Jan. 2020.
[49] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[50] M. J. Pencina, R. B. d'Agostino, R. B. d'Agostino, and R. S. Vasan, "Evaluating the added predictive ability of a new marker: From area under the ROC curve to reclassification and beyond," *Statistics in Medicine*, vol. 27, no. 2, pp. 157–172, Jan. 2008.

[51] L. Ding, Z. Z. Fan, and D. L. Chen, “Auto-categorization of HS code using background net approach,” *Procedia Computer Science*, vol. 60, no. 1, pp. 1462–1471, Jan. 2015.
[52] R. D. A. Batista, D. D. S. Bagatini, and R. Frozza, “Classificação Automática de Códigos NCM Utilizando o Algoritmo Naïve Bayes,” *iSys - Brazilian Journal of Information Systems*, vol. 13, no. 3, pp. 4–29, Jun. 2018.
[53] G. Li and N. Li, “Customs classification for cross-border e-commerce based on text-image adaptive convolutional neural network,” *Electronic Commerce Research*, vol. 19, no. 4, pp. 779–800, Dec. 2019.
[54] J. Luppes and F. Hasibi, “Classifying Short Text for the Harmonized System with Convolutional Neural Networks,” 2019.
[55] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” Jan. 2013.
[56] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes, “RMDL: Random Multimodel Deep Learning for Classification,” *ACM International Conference Proceeding Series*, pp. 19–28, May 2018.
[57] M. Heidarysafa, K. Kowsari, D. E. Brown, K. J. Meimandi, and L. E. Barnes, “An Improvement of Data Classification Using Random Multimodel Deep Learning (RMDL),” *International Journal of Machine Learning and Computing*, vol. 8, no. 4, pp. 298–310, Aug. 2018.
[58] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima,” *International Conference on Learning Representations 2017*, pp. 1–16, Sep. 2016, Accessed: Jul. 25, 2020
[59] “US Imports - Automated Manifest System (AMS) Shipments 2018-2020”, 2020. https://aws.amazon.com/marketplace/pp/US-Imports-Automated-Manifest-System-AMS-Shipments/prodview-stk4wn3mbhx24%23overview (accessed Jun. 24, 2020).