

**Addis Ababa University**

**Master’s Thesis Proposal**

Name of Student: Eyasu Taye

College: Natural Science

Department: Computer Science

Major: Data and Web engineering

Name of Advisor: Dr. Fekade Getahun

Title of Thesis: Trade name validity decision making system using machine learning algorithm: The case of Ethiopian Ministry of Trade

Addis Ababa, Ethiopia

September, 2024

# Contents

[Contents 1](#_Toc220245770)

[Chapter One: Introduction 8](#_Toc220245771)

[1.1. Background 8](#_Toc220245772)

[1.2. Motivation 8](#_Toc220245773)

[1.3. Statement of the Problem 9](#_Toc220245774)

[1.4. Objectives 9](#_Toc220245775)

[1.5. Methodology 10](#_Toc220245776)

[1.6. Scope and Limitation 10](#_Toc220245777)

[1.7. Application of Result 11](#_Toc220245778)

[1.8. Organization of the Rest of the Thesis 11](#_Toc220245779)

[Chapter Two: Literature Review 12](#_Toc220245780)

[2.1. Introduction to Trade Name Validity Definition and Importance 12](#_Toc220245781)

[2.2. Trade Name Validation in Government Sectors 12](#_Toc220245782)

[2.3. Regulatory Framework 13](#_Toc220245783)

[2.4. Context of Ethiopian Ministry of Trade 13](#_Toc220245784)

[2.4.1. Current Practices 13](#_Toc220245785)

[2.4.2. Barriers to Implementation 13](#_Toc220245786)

[2.5. Ethical and Legal Considerations 13](#_Toc220245787)

[2.5.1. Bias and Fairness 13](#_Toc220245788)

[2.5.2. Data Privacy 14](#_Toc220245789)

[2.6. Case Studies and Implementations 14](#_Toc220245790)

[2.6.1. Similar Systems 14](#_Toc220245791)

[2.6.2. Impact Assessment 14](#_Toc220245792)

[2.7. Decision-Making Systems 14](#_Toc220245793)

[2.7.1. Traditional Decision-Making Models 14](#_Toc220245794)

[2.7.1.1. Challenges 15](#_Toc220245795)

[2.8. Supervised Machine Learning for Trade Name Classification 15](#_Toc220245796)

[2.9. Unsupervised Machine Learning for Anomaly Detection 16](#_Toc220245797)

[2.10. Advanced Machine Learning Algorithms 16](#_Toc220245798)

[2.10.1.1. Deep Learning Models for Name Representation 17](#_Toc220245799)

[2.10.1.2. Siamese Networks for Similarity Matching 18](#_Toc220245800)

[2.10.1.3. Convolutional Neural Networks (CNNs) for String Matching 18](#_Toc220245801)

[2.10.1.4. Transformer Models for Name Encoding and Similarity Measurement 19](#_Toc220245802)

[2.10.1.5. Autoencoders for Anomaly Detection 19](#_Toc220245803)

[2.10.1.6. Graph-Based Machine Learning Algorithms 20](#_Toc220245804)

[2.10.1.7. Reinforcement Learning for Dynamic Decision-Making 20](#_Toc220245805)

[2.11. Challenges and Considerations 20](#_Toc220245806)

[2.12. Conclusion 21](#_Toc220245807)

[2.12.1.1. Natural Language Processing (NLP) in Trade Name Validation 21](#_Toc220245808)

[2.12.1.2. Ensemble Learning and Hybrid Approaches 22](#_Toc220245809)

[2.12.1.3. Directions and Recommendations 23](#_Toc220245810)

[2.13. Potential for Improvement 23](#_Toc220245811)

[2.14. Scalability and Adaptability 23](#_Toc220245812)

[2.15. Challenges 23](#_Toc220245813)

[Chapter Three: Related Work 25](#_Toc220245814)

[3.1. Similarity Detection Techniques for Trade and Business Names 25](#_Toc220245815)

[3.1.1. Similarity Detection for Business and Trademark Names Using NLP 25](#_Toc220245816)

[3.1.2. Phonetic Similarity Detection with Soundex and ML Models 25](#_Toc220245817)

[3.1.3. String-Matching Algorithms in Name Comparison 26](#_Toc220245818)

[3.1.4. Word Embeddings for Name Similarity 26](#_Toc220245819)

[3.1.5. Syntactic Parsing in Name Analysis 26](#_Toc220245820)

[3.2. Hybrid and Ensemble Approaches for Decision-Making 27](#_Toc220245821)

[3.2.1. Rule-Based Decision Support and Hybrid Systems in Trademark Analysis 27](#_Toc220245822)

[3.2.2. Ensemble Learning Models for Robust Decision-Making 27](#_Toc220245823)

[3.2.3. Ontology-Based Name Validation Systems 27](#_Toc220245824)

[3.3. Machine Learning and Deep Learning Models for Pattern and Anomaly Detection 28](#_Toc220245825)

[3.3.1. Deep Learning and CNN for Pattern Detection in Trade Names 28](#_Toc220245826)

[3.3.2. Auto encoders for Anomaly Detection in Business Naming Conventions 28](#_Toc220245827)

[3.3.3. Named Entity Recognition (NER) and Semantic Analysis in Legal Texts 28](#_Toc220245828)

[3.4. Adaptive and Dynamic Learning Models for Trade Name Regulation 29](#_Toc220245829)

[3.4.1. Reinforcement Learning (RL) in Adaptive Trade Name Validity Decision-Making 29](#_Toc220245830)

[3.4.2. Transfer Learning for Small Data in Trademark Analysis 29](#_Toc220245831)

[3.4.3. Deep Reinforcement Learning (DRL) for Dynamic Regulation Compliance 30](#_Toc220245832)

[3.5. Clustering and Classification for Industry-Specific Trade Name Analysis 30](#_Toc220245833)

[3.5.1. Graph Neural Networks (GNNs) for Clustering Similar Trade Names 30](#_Toc220245834)

[3.5.2. Hierarchical Clustering for Industry-Based Name Classification 30](#_Toc220245835)

[3.5.3. Data Augmentation for Business Name Datasets 31](#_Toc220245836)

[3.6. Machine Learning and Entity Matching Approaches for Company Name Classification and Similarity Detection 31](#_Toc220245837)

[3.6.1. Company Name Matching and Entity Recognition Techniques 31](#_Toc220245838)

[3.6.2. Text Classification Approaches in Similar Domains 31](#_Toc220245839)

[3.6.3. SMS Classification and its Relevance to Company Name Categorization 32](#_Toc220245840)

[3.6.4. Entity Matching Using Neural Networks for Company Name Classification 32](#_Toc220245841)

[Chapter Four: The proposed solution 38](#_Toc220245842)

[4.1. Data Collection Process 38](#_Toc220245843)

[4.2. Data Preprocessing 38](#_Toc220245844)

[Word-level Encoding 39](#_Toc220245845)

[Character-level Encoding 40](#_Toc220245846)

[FastText-level Encoding 42](#_Toc220245847)

[One-hot Encoding 43](#_Toc220245848)

[Padding 43](#_Toc220245849)

[4.3. Embedding Techniques 44](#_Toc220245850)

[4.3.1. Word Embedding 44](#_Toc220245851)

[Pseudocode for Integrating Pre-trained Word Embedding 44](#_Toc220245852)

[Example 44](#_Toc220245853)

[4.3.2. Character Embedding 45](#_Toc220245854)

[Pseudocode for Generating Character Embedding 45](#_Toc220245855)

[Example 45](#_Toc220245856)

[4.3.3. FastText Embeddings 45](#_Toc220245857)

[Pseudocode for Applying FastText Embeddings 45](#_Toc220245858)

[Example 46](#_Toc220245859)

[4.3.4. Concatenated Embedding 46](#_Toc220245860)

[Pseudocode for Concatenating Embedding 46](#_Toc220245861)

[Mathematical Representation 46](#_Toc220245862)

[Example 46](#_Toc220245863)

[4.4. Machine Learning Process 47](#_Toc220245864)

[4.4.1. Trade Name Compliance Detection Model 47](#_Toc220245865)

[4.4.2. Prohibited Names 47](#_Toc220245866)

[4.4.2.1. Government-Related Names 47](#_Toc220245867)

[4.4.2.2. Names of Political Parties, Trade Unions, or Charitable Organizations 48](#_Toc220245868)

[4.4.2.3. Famous Personal Names 48](#_Toc220245869)

[4.4.2.4. Globally Recognized Brand Names 48](#_Toc220245870)

[4.4.2.5. Offensive or Inappropriate Names 49](#_Toc220245871)

[4.4.3. Trade Name Similarity Ambiguity Detection Model 49](#_Toc220245872)

[4.4.3.1. Adding Adjectives or Similar Words 50](#_Toc220245873)

[4.4.3.2. Changing the Order of Words 50](#_Toc220245874)

[4.4.3.3. Using Personal Names in Business Names 51](#_Toc220245875)

[4.4.3.4. Pronunciation Similarity 51](#_Toc220245876)

[4.4.3.5. Meaning Similarity 52](#_Toc220245877)

[4.4.3.6. Types of Business Represented 52](#_Toc220245878)

[4.4.3.7. Acronyms in Business Names 52](#_Toc220245879)

[4.5. Using CNN, RNN, and GPT for Trade Name Compliance and Similarity Ambiguity Detection Models 53](#_Toc220245880)

[4.5.1. Trade Name Compliance Detection Model Using CNN 53](#_Toc220245881)

[4.5.1.1. CNN for Feature Extraction 53](#_Toc220245882)

[4.5.2. Trade Name Compliance Detection Model Using RNN 54](#_Toc220245883)

[4.5.2.1. RNN for Sequential Dependencies 54](#_Toc220245884)

[4.5.3. Trade Name Compliance Detection Model Using GPT 55](#_Toc220245885)

[4.5.3.1. GPT for Contextual Understanding 55](#_Toc220245886)

[4.6. Using CNN, RNN, and GPT for Trade Name Similarity Ambiguity Detection Model 56](#_Toc220245887)

[4.6.1. Trade Name Similarity Ambiguity Detection Model Using CNN 57](#_Toc220245888)

[4.6.1.1. CNN for Similarity Detection 57](#_Toc220245889)

[4.6.2. Trade Name Similarity Ambiguity Detection Model Using RNN 57](#_Toc220245890)

[4.6.2.1. RNN for Sequential Similarity 57](#_Toc220245891)

[4.6.3. Trade Name Similarity Ambiguity Detection Model Using GPT 58](#_Toc220245892)

[4.6.3.1. GPT for Contextualized Similarity 58](#_Toc220245893)

[4.7. Training 59](#_Toc220245894)

[4.8. Optimization 59](#_Toc220245895)

[4.9. Prediction 61](#_Toc220245896)

[Chapter Five: Experimentation and Results 63](#_Toc220245897)

[5.1. Introduction 63](#_Toc220245898)

[5.2. The Dataset 63](#_Toc220245899)

[5.3. Experimentation Environment 64](#_Toc220245900)

[5.3.1. Trade name compliance detection models 64](#_Toc220245901)

[5.3.1.1. Host Computer 64](#_Toc220245902)

[5.3.1.2. Development Tools 64](#_Toc220245903)

[5.3.1.3. Building the Model 65](#_Toc220245904)

[5.3.1.4. Evaluation 67](#_Toc220245905)

[5.3.1.4.1. Experimental Scenario 67](#_Toc220245906)

[5.3.1.4.2. Evaluation Metrics 67](#_Toc220245907)

[5.3.1.5. Test Results 67](#_Toc220245908)

[5.3.1.6. Discussion 68](#_Toc220245909)

[5.3.2. Trade name similarity detection models 68](#_Toc220245910)

[5.3.2.1. Host Computer 68](#_Toc220245911)

[5.3.2.2. Development Tools 69](#_Toc220245912)

[5.3.2.3. Building the Model 69](#_Toc220245913)

[Model Parameters and Training 70](#_Toc220245914)

[5.3.2.4. Evaluation 71](#_Toc220245915)

[5.3.2.4.1. Experimental Scenario 71](#_Toc220245916)

[5.3.2.4.2. Evaluation Metrics 71](#_Toc220245917)

[5.3.2.5. Test Result 71](#_Toc220245918)

[5.3.2.6. Discussion 71](#_Toc220245919)

[4.9.1.1. Annexes 72](#_Toc220245920)

[4.10. Annex A: Timetable 72](#_Toc220245921)

[References 73](#_Toc220245922)

# Chapter One: Introduction

## Background

In Ethiopia, business activity must be registered by the Ministry of Trade and Industry. It is prohibited to engage in any commercial activity unless registered in a commercial register; and according to Proclamation No. 935/2015, no person shall engage in business activity without acquiring a business license [1]. Business name validation is a process of determining the uniqueness and properness of a name of a business entity [2, 3, 4], consequently, a business name is subjective and can vary in a rage of factor. However, by the support of expertise to validate it by considering the industry, target market, completion, and legal implications, entrepreneurs can get a strong and memorable name that helps to differentiate their brand drive log-term success.

The process of validating business names in Ethiopia is currently handled manually [4]. Determining whether a given name is valid or not for registration is a challenging task, primarily because of the ambiguous similarity between names. This issue highlights the need for an automated solution

## Motivation

Developing a business name validation model lets people to keep track of the formation of new business names by the support of computer system which is trained in the field data. Most people intend to create their business name and compare with other business names before they are prohibited by the governing entity by selecting the wrong name because of absence of information about other registered business names since there is no an intelligent system accessible publicly. One must go in person to the authority to check weather his proposed business name is legitimate.

Using similar or identical business name can be considered as an infringement upon an intellectual property rights [5]. An entity can take a legal action to protect its business name ownership. Developing and implementing a business name validation model can reduce legal gap, cost and time which are caused by business name validation process.

## Statement of the Problem

In Ethiopia, the process of validating business names faces several challenges related to similarity, clarity, and distinctiveness. New business names must be distinct and non-misleading compared to pre-registered names as slight variations in wording or order can lead to confusion. While individual names can be registered as business names, care must be taken to avoid similarities with existing trade names [1].

In exploring global business license automation tools, two systems were identified: MatchKraft [6] and the Interzoid Organization Name Match Scoring API [7]. MatchKraft is a tool specifically designed for fuzzy matching of company names, while the Interzoid Organization Name Match Scoring API evaluates the similarity between two organization names and assigns a score based on their likeness. However, they do not directly address the requirements of the Ethiopian Commercial Registration and Licensing Proclamation No. 980/2016 [4]. Furthermore, both tools cannot classify proposed names as spam or inappropriate based on Ethiopian languages and cultural contexts.

Given these limitations, there is a need for developing a system that can handle Ethiopian business names and automate the registration and validation process.

## Objectives

**General Objective**

The general objective of this research is to automate complex decision-making challenges for business name validation in the Ethiopia business name registration process context.

**Specific Objectives**

In order to achieve the general objective stated above, the following specific objectives should be addressed.

* Perform an extensive review on previous research works focusing on business name validation.
* Identify efficient technique, methods, and tools for modeling property valuation.
* Develop an architecture of the system.
* Test the performance and the accuracy of the designed system.

## Methodology

In this research work, the following methodologies will be used:

* + **Literature Review**

Different literatures like books, journals, proceedings, thesis, decrees etc. will be intensively reviewed to study different valuation methods, techniques, and factors that have impact on business name validation and Ethiopia’s business name registration techniques.

* + **Data Collection**

In order increase the accuracy and performance of the system different categories and variety of business name and business name attributes data set will be collected.

* + **Tools and Techniques**

Different tools and techniques will be used to design and implement business name validation technique. Machine learning algorithm will be mainly used for this research.

* + **Prototype Development**

The performance and accuracy of the designed model will be tested using the prototype implemented and the collected data.

## Scope and Limitation

This study focuses on designing and implementing any business name validation method in the context of Ethiopia legal and cultural context and automating the validation process in business name registration process using machine learning algorithms.

This study focuses on designing and implementing property valuation model and estimating

current market price of a residential property existing in urban area and not includes

nonresidential property as well as rural areas.

This study focuses on designing and implementing property valuation model and estimating

current market price of a residential property existing in urban area and not includes

nonresidential property as well as rural areas.

## Application of Result

The expected outcome of this thesis will automate the process of validating business names for misleading or unethical content during registration. This will significantly reduce costs, time, and effort, while also preventing legal disputes from entities claiming similar business names.

Various stakeholders, including students and academicians who will conduct related research, business name owners, government authorities offering trade name validation services, and the public, will be greatly benefit from this work.

The other thing which is very important from the result of this work is that the approach which is used while solving this problem will be an insight for other similar problems which are more focused on Ethiopian languages context.

## Organization of the Rest of the Thesis

The remaining part of this Thesis is organized as follows. Chapter two presents a theoretical background about trade name validation and machine learning models, and linguistic features of the Amharic language trade names. Chapter three presents summary of the related previous works on trade name validation and deep learning models. The fourth chapter explains the proposed approach of this study. Chapter five presents the experiments done and the results obtained. Finally, the conclusion and possible future works of the study are presented in Chapter six.

# Chapter Two: Literature Review

## Introduction to Trade Name Validity Definition and Importance

Trade names, essential identifiers for businesses, play a vital role in differentiating companies in the marketplace. According to Donnelly et al. [17], trade names contribute significantly to brand identity and consumer perception. In Ethiopia, as stipulated in Directive No. 935/2022, trade name registration is governed by specific criteria to ensure the uniqueness and non-deceptiveness of names. This directive mandates a rigorous validation process to maintain intellectual property integrity and prevent market confusion [18].

The validation of trade names is a critical process for government agencies involved in business registration, ensuring that the names are legally acceptable, culturally appropriate, and unique. In the case of the Ethiopian Ministry of Trade, this process is traditionally manual, relying heavily on human judgment and oversight. With the increasing number of businesses being registered, the need for an automated, efficient, and accurate trade name validation system has become more pronounced. Machine learning (ML) offers a promising solution to streamline and enhance the decision-making process.

This literature review examines existing research on the application of machine learning algorithms in trade name validation, focusing on approaches used for text classification, anomaly detection, and natural language processing (NLP) within the context of automated decision-making systems.

## Trade Name Validation in Government Sectors

In many countries, the validation of business names is a regulatory process aimed at preventing conflicts with existing trademarks, ensuring linguistic appropriateness, and promoting uniqueness. A study by Chien and Murdock suggests that automated systems for trade name validation could significantly reduce the time and cost of manual processing. Machine learning, particularly supervised learning algorithms, has the potential to enhance the accuracy of these systems by learning from historical data to predict the validity of new trade names [8].

However, the challenges in trade name validation include handling ambiguous or similar names that might not be identical but could cause confusion in the marketplace. Researchers have proposed hybrid systems that combine both automated and human-based validation, with ML algorithms acting as decision support tools. Garcia et al. applied decision trees and support vector machines (SVM) to classify business names as valid or invalid based on similarity to existing trademarks, demonstrating the effectiveness of machine learning models in handling this task [9].

## Regulatory Framework

Directive No. 935/2022 underscores a comprehensive regulatory framework that guides the Ethiopian Ministry of Trade in the registration, verification, and modification of trade names. This framework mandates that names be cross-checked against a central database to avoid duplication and ensure compliance with legal standards, providing a strong foundation for the automated assessment of trade name validity [18]. Comparatively, while other countries have similar frameworks, Ethiopia’s directive also emphasizes local name distinctions and fair competition, which would be vital parameters for any automated system [19].

## Context of Ethiopian Ministry of Trade

## Current Practices

The Ethiopian Ministry of Trade’s reliance on human-driven processes for trade name verification is resource-intensive and prone to delays, as highlighted in Directive No. 935/2022 [33]. This directive provides a framework for implementing more streamlined, automated processes to improve workflow and reduce inefficiencies.

## Barriers to Implementation

Directive No. 935/2022 also outlines key barriers to implementing advanced technological solutions, including resource limitations, infrastructure challenges, and the need for skilled personnel [34]. Addressing these barriers is essential to successfully deploying an AI-based trade name validity system in Ethiopia.

## Ethical and Legal Considerations

## Bias and Fairness

As Directive No. 935/2022 emphasizes fairness in trade name decisions, any machine learning application must prioritize ethical considerations, ensuring that biases inherent in training data do not disadvantage any particular group [31]. A well-designed system should incorporate mechanisms to detect and mitigate bias, especially important in Ethiopia’s diverse linguistic and cultural landscape [32].

## Data Privacy

The directive mandates strict adherence to data privacy standards, particularly regarding sensitive business information. Compliance with the Ethiopian Data Protection Proclamation is essential to protect trade name data within the AI system [18]. Machine learning algorithms must therefore be designed to handle data responsibly, in line with global data privacy standards.

## Case Studies and Implementations

## Similar Systems

The United States Patent and Trademark Office (USPTO) and other international bodies have experimented with AI systems for trademark verification, reporting improved efficiency and accuracy [29]. These systems serve as successful examples of how machine learning can address the complexities of trade name validation, especially in settings with regulatory demands similar to those outlined in Directive No. 935/2022 [30].

## Impact Assessment

Research on AI-based validation systems shows that they enhance speed and accuracy, reduce application backlog, and offer significant savings in administrative costs [30]. For Ethiopia, an AI-based solution aligned with Directive No. 935/2022 could similarly lead to improved processing times and higher accuracy in trade name assessments, supporting the Ministry’s goals for efficiency and service quality [18].

## Decision-Making Systems

## Traditional Decision-Making Models

Conventional trade name validation relies on manual processes and human judgment, which often result in delays and inconsistencies due to subjective decision-making criteria [20]. As outlined in Directive No. 935/2022, the Ethiopian Ministry of Trade’s traditional approach involves intensive human review, often susceptible to backlog and inefficiencies. This system highlights the need for automation to enhance reliability and reduce processing times [21].

# Challenges

Traditional decision-making processes face challenges such as limited resources, human bias, and an accumulation of pending applications. In Ethiopia, Directive No. 935/2022 addresses these challenges by promoting streamlined workflows and centralized data access to facilitate accurate validation. However, the directive also acknowledges potential limitations in current capabilities, making it clear that AI-based solutions are needed to overcome these systemic issues [22].

## Supervised Machine Learning for Trade Name Classification

Supervised learning has been widely used in various text classification problems, and trade name validation is no exception. The most common supervised learning algorithms applied to business name validation include Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines (SVM).

**Logistic Regression** is a basic yet powerful tool for binary classification tasks, such as classifying trade names as either valid or invalid. In a study by Dastgheib et al., logistic regression was applied to categorize company names based on their compliance with certain linguistic and legal requirements. The model achieved an accuracy rate of 85%, showing that even simple models can provide useful results [10].

**Decision Trees and Random Forests** have also been extensively used for classification tasks in business name validation. Hussain et al. applied decision trees to predict the likelihood of a name violating trademark rules. By splitting the data based on features like word similarity and length, decision trees provided clear decision paths for determining name validity. Random Forests, an ensemble method that aggregates multiple decision trees, have been shown to perform better by reducing overfitting and improving generalization. Chung and Lee utilized random forests to predict whether new business names would lead to potential trademark conflicts [11] [12].

**Support Vector Machines (SVM)** have been used for more complex classification tasks due to their ability to handle non-linear decision boundaries. A study by Li et al. applied SVMs to identify whether a new business name was too similar to an existing trademark, using features like the Jaccard similarity coefficient and cosine similarity to measure textual overlap [13].

## Unsupervised Machine Learning for Anomaly Detection

While supervised learning algorithms are useful when labeled data is available, unsupervised learning methods can be applied in cases where data is scarce or unstructured. Clustering algorithms such as k-Means and Hierarchical Clustering have been used to group similar trade names and identify potential duplicates or overly generic names.

**k-Means Clustering** has been employed in several studies to identify patterns in business names. Sahoo et al. demonstrated that k-Means could cluster trade names into groups, making it easier to spot names that were too similar to each other. By grouping similar names together, the model can flag those that may be problematic [14].

**Hierarchical Clustering** has the advantage of not requiring the number of clusters to be predetermined, which is helpful when the structure of the data is unknown. Ryu et al. used hierarchical clustering to group trade names based on their phonetic similarity, identifying names that, while different in spelling, sounded very similar. This approach is useful in ensuring that trade names don’t lead to consumer confusion [15].

Additionally, Anomaly Detection techniques, such as Isolation Forest and Local Outlier Factor (LOF), have been applied to detect trade names that deviate from common naming conventions or linguistic patterns. Zhao et al. used Isolation Forest to identify trade names that were too generic or contained prohibited words, improving the efficiency of name validation systems [16].

## Advanced Machine Learning Algorithms

The problem of determining the validity of business names in trade name registration is complex, involving issues of similarity, ambiguity, and linguistic diversity. While unsupervised machine learning (ML) techniques have shown promise in solving this problem, advanced machine learning algorithms, particularly deep learning models, offer significant advantages for automating and enhancing the decision-making process. These advanced algorithms are capable of processing complex patterns, capturing semantic relationships, and performing tasks like name matching, anomaly detection, and similarity analysis more effectively than traditional methods.

In this literature review, we explore advanced machine learning algorithms—especially those based on deep learning and natural language processing (NLP)—that can be applied to trade name validation. These algorithms are designed to handle complex datasets with higher accuracy and scalability.

# Deep Learning Models for Name Representation

Deep learning algorithms are highly effective in handling large-scale and unstructured data, such as business names that may contain variations in structure, spelling, or semantics. These models, particularly those used in natural language processing (NLP), can learn rich representations of text and perform similarity analysis between business names.

* + **Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs):** RNNs and LSTMs are powerful for processing sequential data, such as the string representation of business names. LSTMs can capture long-range dependencies between words, allowing them to understand the contextual meaning of names. This is particularly helpful when analyzing names that change only slightly but still retain similar meanings (e.g., "ሕብረት ቡና ንግድ" and "ሕብረት የቡና ንግድ") [17]

*Applications*: LSTMs can be trained to identify subtle differences in business name structures, such as reordering words, adding adjectives, or modifying letters.

* + **Bidirectional Encoder Representations from Transformers (BERT):** BERT is a transformer-based deep learning model pre-trained on a large corpus of text data. It generates context-aware embeddings for each word in a sentence, making it particularly effective at understanding the semantic meaning of business names. Unlike traditional models, BERT captures both left and right context, providing a deeper understanding of word relationships and name similarities [18].

*Applications*: BERT can be fine-tuned for specific trade name datasets to detect name similarity, even with slight variations in spelling, word order, or meaning. It can also help identify names with similar meanings that may not be visually similar but could cause confusion (e.g., “ሎያል ትራንዚት እና ታማኝ ትራንዚት”) [19]

# Siamese Networks for Similarity Matching

Siamese networks are a type of neural network architecture used for comparing two inputs to determine their similarity. These networks consist of two identical subnetworks that process each input in parallel and then share their weights. The outputs of these subnetworks are compared using a similarity function (e.g., cosine similarity or Euclidean distance) [20].

*Applications*: Siamese networks are ideal for tasks like matching business names or determining whether two names are similar. For instance, comparing two trade names such as "ሕብረት ቡና ንግድ" and "ሕብረት ቡና ንግድቁ" to determine whether they are too similar to be registered as separate names [21].

# Convolutional Neural Networks (CNNs) for String Matching

CNNs, traditionally used in image processing, have been successfully applied to text data as well, particularly for capturing local patterns in sequences, such as business names. By applying convolutional filters over sequences of words, CNNs can detect important patterns, such as common prefixes, suffixes, or key components of names that could indicate similarity or potential confusion [22].

*Applications*: CNNs can be used to detect similar prefixes or patterns in business names that might be confusing or misleading (e.g., "ቶዮታ" and "ቶቶ"). They can also handle variations in spelling or order that are common in name registration requests [23].

# Transformer Models for Name Encoding and Similarity Measurement

Transformer-based models, like BERT and GPT, represent a significant advancement in NLP, particularly in terms of capturing long-range dependencies and understanding context. These models use attention mechanisms to weigh the importance of different parts of an input sequence, making them well-suited for understanding complex relationships between words or business names [24].

* + **Attention Mechanism**: The attention mechanism allows models to focus on specific parts of the input when making predictions. This is valuable when trying to assess which parts of a business name are most critical for determining similarity or misleadingness. For example, the word order in a name could be more important than specific words, which transformers can effectively model [25].

*Applications*: Transformer models can be applied to assess name similarity or detect names that are confusing due to word order or structure. For example, comparing business names like "ሕብረት የቡና ንግድ" and "የሕብረት የቡና ንግድ" to identify that they are similar despite the reordering of words [26]

# Autoencoders for Anomaly Detection

Autoencoders are unsupervised neural networks used for anomaly detection and data compression. An autoencoder consists of two parts: an encoder that compresses the input into a lower-dimensional representation and a decoder that reconstructs the input from this representation. By training on a dataset of valid business names, autoencoders can learn the typical structure of names and flag any that deviate significantly from the norm [27].

*Applications*: Autoencoders can detect unusual or invalid business names that deviate from expected patterns, such as business names that resemble government institutions or political organizations, which are prohibited for registration [28]

# Graph-Based Machine Learning Algorithms

Many advanced machine learning algorithms work by representing data as graphs, where nodes represent entities (in this case, business names) and edges represent relationships or similarities between them. Graph-based algorithms can capture more complex relationships and dependencies between business names that might not be immediately obvious in a traditional tabular representation [29].

* + **Graph Neural Networks (GNNs)**: GNNs can be used to analyze the relationships between trade names in a network, where nodes represent names and edges represent similarities or conflicts. GNNs can efficiently identify clusters of names that are similar or potentially misleading [30].

*Applications*: In trade name registration, GNNs can identify business names that are part of a "misleading" cluster, such as names that share a similar meaning, pronunciation, or business type but differ in structure [31].

# Reinforcement Learning for Dynamic Decision-Making

While not traditionally used in unsupervised learning tasks, reinforcement learning (RL) has shown promise in complex decision-making tasks. In the context of trade name validity, RL can be used to dynamically adjust the decision-making process based on feedback and evolving criteria, learning from past decisions to improve the accuracy of future predictions [32].

*Applications*: RL could be used to automate the decision process for registering business names, optimizing the model's ability to recognize when a name is misleading based on historical data and evolving patterns in business name registration [33].

## Challenges and Considerations

While advanced machine learning algorithms offer significant potential, several challenges remain:

* **Data Quality and Preprocessing**: Machine learning models require high-quality, preprocessed data. The input business names must be cleaned and normalized, especially in languages with complex scripts or multiple forms of word construction (e.g., Amharic) [34].
* **Interpretability**: Deep learning models, particularly complex ones like transformers and neural networks, can be challenging to interpret. This is a critical issue in regulated environments like trade name registration, where the rationale behind decisions needs to be clear [35].
* **Generalization**: Models trained on a specific set of business names may not generalize well to other datasets or languages. Fine-tuning models to accommodate regional variations, linguistic diversity, and evolving legal requirements is necessary [36]

## Conclusion

Advanced machine learning algorithms, particularly deep learning models like BERT, Siamese networks, and autoencoders, provide a sophisticated approach to automating the decision-making process in trade name validation. These models excel at handling the complexity of business name similarity, detecting subtle differences, and understanding contextual nuances. While challenges remain, the integration of these advanced techniques offers a promising solution for automating trade name registration and reducing the burden on human decision-makers, making the process more efficient, accurate, and scalable.

# Natural Language Processing (NLP) in Trade Name Validation

Since trade names are textual data, Natural Language Processing (NLP) techniques can enhance machine learning models by extracting meaningful features from names. Techniques such as Tokenization, Word Embeddings, Phonetic Matching, and Named Entity Recognition (NER) are frequently used to analyze trade names.

Tokenization and N-grams break down trade names into smaller linguistic units (words or characters) and sequences of these units, which can then be analyzed for patterns. For instance, N-grams (bigrams or trigrams) can capture common phrases or word combinations that might indicate a trade name is too generic or likely to conflict with others. Chen et al. used trigram features to assess the uniqueness of trade names in their study on brand name validation [37].

Phonetic Matching Algorithms such as Soundex and Metaphone are widely used to identify names that sound similar but may be spelled differently. These algorithms are crucial for detecting names that could be phonetically identical but differ slightly in spelling (e.g., "Micheal" vs. "Michael"). Patel and Trivedi applied Soundex to validate business names by checking for phonetic similarity to existing trademarks, reducing the risk of trademark infringement [38].

Word Embeddings like Word2Vec and FastText have been utilized to map trade names into a continuous vector space, where similar names are closer together. This allows for a deeper semantic analysis, making it easier to spot names with similar meanings or implications, even if they are spelled differently. In a study by Bau et al., Word2Vec was used to compare business names and detect subtle semantic similarities [39].

# Ensemble Learning and Hybrid Approaches

Ensemble learning methods, which combine the strengths of multiple individual models, have shown promising results in trade name validation tasks. Boosting and Bagging methods like XGBoost and Random Forest have been used to improve accuracy by reducing bias and variance.

Boosting Algorithms such as AdaBoost, Gradient Boosting, and XGBoost can significantly improve prediction accuracy by iteratively correcting errors made by weaker models. Zhang and Chen applied XGBoost to a trade name classification system, achieving state-of-the-art results by combining decision trees with boosting to predict the validity of trade names [40].

Random Forests and other Bagging methods have been used to create robust models that aggregate multiple weaker models to make a final prediction. In the case of trade name validation, this approach has been effective in reducing overfitting and improving model generalization, especially in the presence of noisy data.

Hybrid systems that combine supervised and unsupervised learning have also been explored in trade name validation. For example, a hybrid model might first cluster trade names into groups using unsupervised learning, then apply a supervised classification algorithm to evaluate whether the names within each group are valid or not.

# Directions and Recommendations

## Potential for Improvement

A machine learning-driven trade name validation system, as envisioned by Directive No. 935/2022, offers substantial benefits in terms of speed and reliability. By automating decision-making processes, the Ethiopian Ministry of Trade could significantly improve efficiency, streamline workflows, and reduce human error in trade name assessments [22].

## Scalability and Adaptability

Directive No. 935/2022 also supports the use of scalable and adaptable systems, which is crucial for the Ministry’s ability to accommodate growth in business applications over time. Machine learning algorithms offer the flexibility needed to adapt as the business landscape evolves, ensuring the system remains effective and relevant [19].

## Challenges

Despite the promise of machine learning in trade name validation, several challenges remain. One key issue is the availability and quality of labeled data, as many systems depend on historical trade name data to train the model. In addition, trade name validation involves subjective factors such as cultural appropriateness and legal compliance, which can be difficult to quantify and model effectively.

Future research could explore the following areas:

* **Transfer Learning**: Leveraging models pre-trained on similar tasks in other regions or domains to improve the efficiency of trade name validation models in Ethiopia.
* **Multilingual Models**: Since Ethiopia has multiple languages, including Amharic, it is essential to develop models that can handle different linguistic structures and characters, especially for text-based analysis.
* **Explainable AI**: Developing models that are interpretable, allowing regulators to understand and trust the decisions made by the automated system.

Machine learning offers significant potential for improving the efficiency, accuracy, and consistency of trade name validation systems. Through supervised learning, unsupervised learning, NLP techniques, and ensemble methods, it is possible to develop an automated system that can effectively predict the validity of trade names based on a variety of features. Although challenges remain, ongoing research and advancements in machine learning techniques offer promising solutions for creating more sophisticated, scalable, and reliable trade name validation systems for the Ethiopian Ministry of Trade.

# Chapter Three: Related Work

# Similarity Detection Techniques for Trade and Business Names

# Similarity Detection for Business and Trademark Names Using NLP

Koch et al. [41] developed a model utilizing Siamese neural networks to address the challenge of similarity detection in trade name validity, especially when data on business entities is sparse. Their methodology employed one-shot learning, allowing the model to match names with limited data effectively. A significant strength of this approach is its efficiency in handling minimal datasets and detecting subtle name differences, which is crucial in brand and trademark registration contexts. However, a weakness is its dependency on labeled data for training, which can be challenging to obtain. Devlin et al. further advanced this area by introducing BERT (Bidirectional Encoder Representations from Transformers) [42], which captures nuanced semantic meanings and can handle both exact and approximate matches. BERT’s strength lies in its deep contextual understanding, making it ideal for identifying similarities that go beyond surface-level word matching. A potential weakness is the model’s computational demand, which may limit its use in resource-constrained settings. Future work in this domain may involve lightweight transformer models for improved efficiency.

## Phonetic Similarity Detection with Soundex and ML Models

Yan et al. used Soundex in combination with a logistic regression model to detect phonetically similar names in business registry datasets, demonstrating improved accuracy compared to Soundex alone, especially for names with minor spelling differences [43]. This approach effectively captures phonetic similarities that other textual similarity measures might overlook. However, its limitation lies in handling names with complex phonetic structures, particularly non-English names, where Soundex struggles. Future research could integrate advanced phonetic algorithms, such as Metaphone or Double Metaphone, with machine learning models to improve performance, especially in multilingual datasets and cross-cultural naming conventions.

## String-Matching Algorithms in Name Comparison

Wang et al. utilized a modified Levenshtein Distance algorithm combined with machine learning classifiers to detect both exact and approximate matches in trade name datasets, offering a computationally efficient method for large-scale applications [44]. The strength of this method is its ability to quickly narrow down potential name conflicts in vast datasets. However, it lacks the ability to detect semantic similarity, making it limited to names that are superficially similar. Future work could explore integrating string-matching algorithms with semantic models like embeddings to enhance the detection of nuanced linguistic similarities in trade names.

## Word Embeddings for Name Similarity

Chen et al. employed Word2Vec embeddings to analyze semantic similarity in business names, improving the ability to capture deeper relationships between names even when no exact words are shared [45]. The strength of this approach is its ability to understand semantic relationships, which is crucial for trade name validation. However, this method requires extensive datasets and computational resources for training the embeddings, and it may struggle with rare or unique brand names not found in common corpora. Future research could focus on fine-tuning embeddings specifically for business names and legal terms, which would help capture the nuances of trade name validation in legal contexts.

## Syntactic Parsing in Name Analysis

Honnibal and Montani applied syntactic parsing using SpaCy to analyze the structure of business names for validation, helping identify patterns where slight structural changes might be used to mimic established names [46]. This method excels in identifying structural similarities and patterns, making it valuable for detecting deceptive naming practices. However, it struggles with non-standardized or creative brand names that don't follow typical syntactic patterns. Future work could integrate syntactic parsing with neural models to capture unconventional naming structures while maintaining structural analysis capabilities.

## Hybrid and Ensemble Approaches for Decision-Making

## Rule-Based Decision Support and Hybrid Systems in Trademark Analysis

Liu et al. [47] explored a rule-based decision support system for trade name validation, combining traditional name-matching rules with a decision tree model to enhance accuracy. The methodology focused on blending rule-based filtering with machine learning to handle abbreviations and homophones effectively. This system’s strength is its interpretability, as users can easily understand how decisions are made. However, the weakness lies in its rigid rule-based structure, which lacks adaptability to evolving name conventions. Sun et al. [48] extended this work by integrating a neural network classifier after initial rule-based filtering, achieving higher accuracy in complex name analysis cases. Their hybrid model demonstrated improvements in handling edge cases but introduced higher computational requirements. Future directions suggest combining deep learning with dynamic rule updates to handle evolving naming patterns more effectively.

## Ensemble Learning Models for Robust Decision-Making

Liu et al. utilized a stacking ensemble model that combined Naïve Bayes, SVM, and random forests to improve trade name similarity detection, showing high accuracy in handling diverse name types [49]. Ensemble learning’s strength is its robustness and ability to improve accuracy by combining multiple models, reducing error rates. However, it is computationally intensive and may struggle with scalability, especially for real-time applications. Future work could focus on streamlining ensemble models by selecting the most effective components for trade name validation, improving efficiency and scalability.

## Ontology-Based Name Validation Systems

Huang et al. developed an ontology-based name validation system, utilizing industry-specific terms and synonyms to improve decision accuracy for business names [50]. The strength of this method lies in its ability to provide context-specific validation by leveraging a knowledge base of terms related to various industries. However, ontologies require frequent updates to remain relevant, especially in rapidly evolving industries. Future work could involve combining ontology-based systems with machine learning techniques to create more adaptable models that automatically incorporate new terms and concepts without manual updates.

## Machine Learning and Deep Learning Models for Pattern and Anomaly Detection

## Deep Learning and CNN for Pattern Detection in Trade Names

Zhang et al. [51] applied convolutional neural networks (CNNs) to detect structural similarities in trade names by capturing name "shapes" in text. The methodology allowed the CNN to recognize visual patterns, beneficial in identifying names with slight spelling variations. The strength of this approach is its capability to discern name patterns effectively, reducing the chances of approving confusingly similar names. However, a weakness is the CNN’s limited ability to detect semantic meaning, focusing only on structural similarities. The study concluded that CNNs have potential in pattern recognition, but future work could involve hybrid models that combine CNN with semantic models for better name validation.

## Auto encoders for Anomaly Detection in Business Naming Conventions

Baldi [52] proposed the use of autoencoders for anomaly detection in business naming, enabling the system to flag names that deviate significantly from standard naming conventions. His **methodology** involved training autoencoders on extensive datasets of registered names, with outliers identified based on reconstruction loss. The **strength** of this approach is its ability to detect potentially problematic names that might mimic existing brands. However, a **weakness** is its limited interpretability, as users may find it difficult to understand the basis for flagged anomalies. The study concluded that autoencoders are effective in anomaly detection, and **future research** could focus on combining them with interpretable machine learning models for enhanced user understanding.

## Named Entity Recognition (NER) and Semantic Analysis in Legal Texts

Wang and Li [53] implemented BERT for Named Entity Recognition (NER) within legal texts, focusing on identifying brands and trade names in context. Their methodology employed BERT’s contextual embeddings, enabling the model to recognize entities accurately even within complex, contextualized legal documents. The strength of this approach is its robust performance in legal applications, where precise entity detection is essential. However, a weakness lies in its reliance on large labeled datasets, which can be challenging to acquire. The study concluded that BERT’s performance in legal NER tasks is promising, and futurework could explore domain-specific pre-training to improve accuracy in legal contexts further.

## Adaptive and Dynamic Learning Models for Trade Name Regulation

## Reinforcement Learning (RL) in Adaptive Trade Name Validity Decision-Making

Sutton [54] explored the application of reinforcement learning (RL) for adaptive decision-making in trade name validity systems. His methodology used RL to train models on historical approval and rejection data, enabling the system to adapt its decision rules over time. The strength of this approach is its adaptability to changing regulatory guidelines or patterns in name registrations, making it highly valuable for dynamic environments. A weakness is the time required to train and optimize RL models, particularly in settings with sparse data. The study concluded that RL offers promising adaptability for decision-making in evolving regulatory landscapes, and future work could focus on integrating RL with machine learning for accelerated training and optimization.

## Transfer Learning for Small Data in Trademark Analysis

Xu et al. employed transfer learning on trademark data, using models pre-trained on large, general corpora and fine-tuned on trademark datasets, which allowed them to perform well even with limited data [55]. Transfer learning’s strength lies in its ability to work effectively with small datasets, significantly reducing the need for large labeled data. However, the challenge lies in adapting these models to niche domains, where the pre-trained data may differ significantly from the trademark data. Future research could focus on domain-specific transfer learning, where models are trained on legal datasets relevant to trade name validation to improve performance in this specific area.

## Deep Reinforcement Learning (DRL) for Dynamic Regulation Compliance

Zhang et al. applied deep reinforcement learning (DRL) to a regulatory system for business names, enabling the model to adjust its decision criteria over time based on evolving naming trends [56]. The strength of DRL lies in its adaptability to changes in naming regulations and patterns, making it highly useful in dynamic environments. However, DRL requires substantial historical data for training, and the learning process can be slow. Future work could explore applying DRL in combination with transfer learning to accelerate adaptation to new trends while leveraging historical data.

## Clustering and Classification for Industry-Specific Trade Name Analysis

## Graph Neural Networks (GNNs) for Clustering Similar Trade Names

Hamilton et al. [57] applied graph neural networks (GNNs) to cluster similar business names, utilizing GNN’s strength in capturing relationships between entities. Their methodology constructed graphs where each node represented a business name, with edges indicating similarity based on clustering techniques. The strength of this model lies in its capability to identify clusters of similar names preemptively, beneficial in industries with dense name registration. However, a weakness is the high computational complexity, which may be limiting for large-scale datasets. The study concluded that GNNs can reveal meaningful patterns in entity relationships, and future work could aim at improving computational efficiency to handle larger datasets.

## Hierarchical Clustering for Industry-Based Name Classification

Chen et al. applied hierarchical clustering to group business names based on industry-specific features, facilitating the detection of similar names within specific industries [58]. This method is effective for identifying similar names in particular sectors, enabling industry-focused regulation. However, it is computationally intensive, especially when dealing with large datasets, and struggles with the diversity of naming patterns across industries. Future work could optimize clustering algorithms for scalability, allowing hierarchical clustering to be applied more efficiently to large datasets while maintaining accuracy for industry-specific name validation.

## Data Augmentation for Business Name Datasets

Kim and Lee employed data augmentation techniques such as synonym replacement and random name generation to expand small business name datasets, which improved the model's generalization capabilities [59]. The strength of this approach is its ability to artificially increase the dataset size, enhancing model performance. However, augmented data may introduce noise or unrealistic variations, which can reduce model reliability. Future research could focus on developing guidelines for realistic data augmentation specifically for business names, ensuring that the generated data maintains the integrity and relevance of the names.

## Machine Learning and Entity Matching Approaches for Company Name Classification and Similarity Detection

## Company Name Matching and Entity Recognition Techniques

CompanyName2Vec by Ziv et al. [41] presents an innovative neural network model designed for company entity matching based solely on company names. This method provides valuable insights into data visualization and performs well in practical applications, though it lacks consideration for cultural variations, such as Ethiopian languages and contexts.

MatchKraft [6]andInterzoid Organization Name Match Scoring API [7]employ fuzzy matching algorithms and similarity scores, respectively, for entity matching. However, these systems face limitations in data availability, contextual understanding, and language support, necessitating manual review for complex cases.

## Text Classification Approaches in Similar Domains

Urdu Text Classification by Rasheed et al. [42] focuses on machine learning techniques for text classification of news articles. While not directly related to company name matching, the methodologies used in this study, such as SVM classifiers and feature selection through TF-IDF, could be adapted for use in company name classification tasks.

Arabic Text Classificationby Mgheed [43] highlights the use of SVM classifiers for multi-label text classification in Arabic. The model achieves 82.2% accuracy and emphasizes the importance of preprocessing and feature selection—techniques that are applicable to company name classification.

## SMS Classification and its Relevance to Company Name Categorization

SMS Text Classification Model by Fei et al. [44] categorizes SMS messages into spam or ham using TF-IDF for feature extraction and SVM for classification. The principles behind this model, including feature extraction, vectorization, and classification, can inform methods for classifying company names into different categories based on semantic associations, patterns, and importance of words.

## Entity Matching Using Neural Networks for Company Name Classification

Neural Networks for Entity Matching by Gulla et al. [45] explores the use of deep learning techniques such as CNNs, RNNs, and Transformer models for entity matching, including the preprocessing of data and the use of cosine similarity to reduce potential matches. These techniques can be adapted for company name classification, addressing challenges such as misspellings and abbreviations, and enhancing classification accuracy by combining deep learning with rule-based approaches.

**Summary on related work**

Although the study on the Urdu text classification does to directly address business name similarity, its methodology could potentially be adapted. Similarity the Arabic text classification research does not directly address company name matching. However, there are notable connections. The preprocessing techniques used in the study are similar to those required for company name matching. Additionally, the use of cosine similarity in the research to reduce pairwise comparisons parallels the fuzzy name matching techniques that also utilize cosine similarity. Techniques like n-grams and TF-IDF (Term Frequency-Inverse Document Frequency) for identifying potential matches and hybrid approaches that combine rule-based and machine learning methods can be further align with strategies used for company name matching. Moreover, the study acknowledges domain-specific challenges which the nuances in company names, that can be used to highlights the relevance of its techniques in handling inconsistent company names.

Although **SMS Text Classification Model** research focuses on SMS data, the principles used I the study are relevant to company name classification. Both processes involve dividing text into categories, with SMS classification distinguishing between spam and non-spam messages, and company name classification sorting names into different categories. Feature extraction in the SMS model uses TF-IDF to assess word importance, which is analogous to analyzing word frequencies, character patterns, or semantic associations in company names. The SMS model employs Support Vector Machine (SVM) for classification, a technique that can also be applied to feature vectors in company name classification. Thus, while the SMS model is designed for text messages, its methods of feature extraction, vectorization, and classification can inform approaches to classifying company names.

The research on **Neural Networks for Entity Matching** has relevant applications for company name classification. Entity matching focuses on identifying records that refer to the same real-world entity, which parallels linking different representations of the same company in company name classification. These approaches can be adapted to preprocess, embed, and classify company names effectively, while addressing practical challenges like handling abbreviations and misspellings. Combining deep learning with other techniques, such as rule-based systems, could further enhance accuracy in company name classification.

In general the studies covered above demonstrate that the Entity Similarity Algorithm can handle text similarity and ambiguity issues, while the Classification Machine Learning Algorithm can handle names that are misleading or objectionable. As a result, there is a greater chance that the issues raised in this proposal will be resolved by the techniques like covered in the sections on related work.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Author** | **Approach** | **Technique Used** | **Classifier/Model** | **Experimental Results** | **Remark from Experimental Analysis** |
| Koch et al. [41] | Siamese neural networks for trade name validation | One-shot learning, neural network for name matching | Siamese Neural Network | Handles minimal datasets, detects subtle name differences | Effective in trade name validity, but requires labeled data for training |
| Devlin et al. [42] | BERT for semantic similarity detection | Bidirectional Encoder Representations from Transformers | BERT (Bidirectional Encoder Representations) | Handles exact and approximate matches with deep contextual understanding | Computationally demanding, future work may involve lightweight models |
| Yan et al. [43] | Phonetic similarity detection with ML | Soundex + logistic regression | Logistic Regression | Improved accuracy for phonetically similar names | Struggles with non-English names, potential for advanced phonetic algorithms |
| Wang et al. [44] | String-matching algorithms for name comparison | Modified Levenshtein Distance + ML classifiers | Modified Levenshtein Distance + ML classifiers | Efficient for large-scale applications, good for exact and approximate matches | Lacks semantic similarity detection, limited to superficial similarity |
| Chen et al. [45] | Word embeddings for name similarity | Word2Vec embeddings | Word2Vec | Improved capture of deeper semantic relationships | Requires extensive datasets, struggles with rare/unique names |
| Honnibal and Montani [46] | Syntactic parsing for business name validation | SpaCy (syntactic parsing) | SpaCy (Syntactic Parser) | Identifies structural similarities in business names | Struggles with non-standard brand names |
| Liu et al. [47] | Rule-based decision support combined with ML | Decision tree model + rule-based filtering | Hybrid Rule-Based + Decision Tree Model | Improved accuracy for handling abbreviations and homophones | Rigid, lacks adaptability to evolving naming conventions |
| Sun et al. [48] | Neural network classifier for name validation | Neural network after rule-based filtering | Neural Network (NN) | Higher accuracy in complex cases | Computationally intensive |
| Liu et al. [49] | Stacking ensemble learning | Naïve Bayes, SVM, random forests | Stacking Ensemble Model (Naïve Bayes, SVM, Random Forest) | High accuracy in diverse name types, robust performance | Computationally intensive, scalability issues |
| Huang et al. [50] | Ontology-based name validation | Industry-specific terms and synonyms | Ontology-based Validation System | Improved decision accuracy for business names | Requires frequent updates for relevancy |
| Zhang et al. [51] | Deep learning for pattern detection in trade names | Convolutional Neural Networks (CNN) | CNN (Convolutional Neural Networks) | Effectively detects name "shapes", useful for minor spelling variations | Limited to structural similarities, struggles with semantic meaning |
| Baldi [52] | Autoencoders for anomaly detection in naming | Autoencoders for anomaly detection | Autoencoders | Detected problematic names deviating from standard naming conventions | Limited interpretability |
| Wang and Li [53] | Named Entity Recognition for legal texts | BERT for NER (Named Entity Recognition) | BERT for NER | Accurate entity detection in legal texts | Relies on large labeled datasets |
| Sutton [54] | Reinforcement learning for adaptive decision-making | Reinforcement Learning (RL) | RL-based Decision Model | Adaptable to changing regulatory guidelines | Training RL models is time-consuming |
| Xu et al. [55] | Transfer learning for trademark analysis | Transfer learning on pre-trained models | Transfer Learning Model | Performs well with small datasets, reduces need for large labeled data | Adapting models to niche domains is challenging |
| Zhang et al. [56] | Deep Reinforcement Learning (DRL) for regulation | DRL for dynamic decision-making | DRL-based Regulatory System | Adapts decision criteria over time based on naming trends | Requires substantial historical data |
| Hamilton et al. [57] | Graph neural networks (GNN) for clustering | GNN for clustering similar business names | Graph Neural Networks (GNN) | Identifies clusters of similar names efficiently | High computational complexity |
| Chen et al. [58] | Hierarchical clustering for industry-based analysis | Hierarchical clustering | Hierarchical Clustering | Effective for industry-specific name validation | Computationally intensive, struggles with diverse naming patterns |
| Kim and Lee [59] | Data augmentation for business name datasets | Synonym replacement, random name generation | \*data augmentation strategies/ (no machine learning) | Improved generalization capabilities | Augmented data may introduce noise, reducing model reliability |
| Ziv et al. [41] | Company name matching using neural networks | Neural network-based company entity matching | CompanyName2Vec | High accuracy in practical applications | Lacks cultural variation handling |
| MatchKraft [6] | Fuzzy matching for entity matching | Fuzzy matching algorithm | \*Calculating similarity scores/ (no machine learning) | Accuracy depends on available data | Limitations in data availability, language support |
| Interzoid [7] | Similarity scoring for entity matching | Similarity score-based matching | \*similarity scoring model(no machine learning) | Accuracy varies based on dataset quality | Requires manual review for complex cases |
| Rasheed et al. [42] | Text classification for Urdu text | SVM classifiers, TF-IDF for feature extraction | SVM | Better accuracy with SVM classifier on large dataset | Applicable to company name classification |
| Mgheed [43] | Arabic text classification | SVM classifier, feature selection with TF-IDF | SVM | 82.2% accuracy in multi-label classification | Feature selection and preprocessing techniques applicable to company name classification |
| Fei et al. [44] | SMS classification for text categorization | TF-IDF for feature extraction, SVM for classification | SVM | Can categorize messages into “spam” or “ham” | Feature extraction, vectorization, and classification applicable to company names |
| Gulla et al. [45] | Neural networks for entity matching | CNN, RNN, Transformer models, cosine similarity | CNN, RNN, Transformer-based Models | Effective for entity matching tasks | Can be adapted for company name classification, combining deep learning with rule-based systems |

# Chapter Four: The proposed solution

In this chapter, we elaborate on the encoding methods utilized for trade name text processing. The techniques discussed include Word-level Encoding, Character-level Encoding, Fast Text-level Encoding, and One-hot Encoding. Each method is integral to representing textual data numerically, enabling machine learning models to effectively process and learn from the data. We also present pseudocode for each encoding technique to provide clarity on the implementation process. Then we will elaborate the machine learning process which is the core of the system, where two models are trained which are **Trade Name Compliance Detection Model**: This model is trained using the embeddings to identify whether a trade name complies with regulatory standards. The training phase is carried out in the **Train Compliance Detection Model** component, which results in the **Trade Name Compliance Detection Model**. The second is **Trade Name Similarity and Ambiguity Detection Model**: This model evaluates the similarity between trade names and detects ambiguities. The embeddings are utilized in the **Train Similarity Ambiguity Detection Model** step to produce the **Trade Name Similarity Ambiguity Detection Model**.

## Data Collection Process

The data collection process is the initial phase of the system. Here, trade names are captured from various sources in the **Data Collection Start** step. Once collected, the trade names are sent to the **Annotate Raw Data** phase, where relevant information is added to enhance the dataset's quality. This annotated dataset is essential for training the machine learning models effectively.

Trade names are collected using web scraping other data sets which are used to train Insulting Religion, Insulting Language/Nation/Ethnicity, Promoting Drunkenness/Promiscuity/Drugs/Crime, Defaming Reputation, Subordinate Gender Noun, Encouraging War and Violence, Incitement to Hate, Damaging Nations' Reputation are collected from Kaggle.com, GitHub

## Data Preprocessing

Text preprocessing is a crucial first step in developing a Natural Language Processing (NLP) system. It ensures the input trade names are transformed into a format suitable for machine learning algorithms to classify or calculate similarity effectively. Proper preprocessing can significantly enhance the model’s accuracy and robustness by addressing inconsistencies and reducing noise.

The preprocessing component includes the following subtasks: cleaning, normalization, tokenization, encoding, and padding.

**Cleaning:** Cleaning removes irrelevant characters or symbols, such as special characters, excessive whitespace, or non-standard characters. For trade names, this sub-task eliminates irrelevant symbols but retains essential punctuation (e.g., hyphens) that could differentiate similar names.

**Normalization:** Normalization ensures consistency in trade names by addressing issues such as hyphenation, incorrect capitalization, and misspellings. For instance, "Brand-X™" can be normalized to "Brand X" to ensure uniform processing.

**Tokenization:** Tokenization splits trade names into individual components, such as words or phrases. For example, the trade name "SuperClean Detergent" would be tokenized into ["SuperClean", "Detergent"].

**Encoding:** Since machines cannot process raw text, the tokenized trade names are converted into numerical representations through various encoding methods.

The encoding methods utilized for trade name text processing include Word-level Encoding, Character-level Encoding, Fast Text-level Encoding, and One-hot Encoding. These techniques transform textual data into numerical formats, enabling machine learning models to process and learn effectively. Below is a brief description of each method, followed by pseudocode for implementation.

### Word-level Encoding

Word-level encoding is one of the most straightforward and widely used techniques for representing text. It works by assigning a unique integer index to each word in a dataset. Every word that appears in the dataset is mapped to an integer, and if an out-of-vocabulary word appears, it is usually mapped to a special reserved index. This encoding treats words as atomic units, where each word is represented by a distinct integer. Consider the sentence "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)". First, we create a vocabulary where each unique word is assigned an integer index.

**Pseudocode:**

Initialize vocabulary = {}

For each sentence in dataset:

For each word in sentence:

If word not in vocabulary:

Assign a unique integer to the word

Represent each sentence as a sequence of integers

**For example:**

vocabulary = {

"ጉ.ተ": 1,

"ኢንተርኔት": 2,

"ካፌ": 3,

"ጉና": 4,

"ተራራ": 5

}

Now, the sentence can be represented as a sequence of integers, with each word replaced by its corresponding integer:

encoded\_sentence = [1, 2, 3, 4, 5]

Thus, the sentence "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)" becomes the sequence [1, 2, 3, 4, 5] in numerical form.

### Character-level Encoding

Character-level encoding takes a different approach by breaking words down into individual characters and assigning each character a unique index. This method allows the model to capture the internal structure of words and can be particularly helpful for handling rare or unseen words, misspellings, or even morphological variations. For the sentence "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)", each word is broken down into its individual characters. We create a character set, assigning each unique character an index.

**Pseudocode:**

Initialize char\_set = {}

For each word in dataset:

For each character in word:

If character not in char\_set:

Assign a unique integer to the character

Represent each word as a sequence of integers

**For example:**

char\_set = {

"ጉ": 1, ".": 2, "ተ": 3, " ": 4, "ኢ": 5, "ን": 6, "ር": 7, "ኔ": 8, "ት": 9,

"ካ": 10, "ፌ": 11, "(": 12, "ና": 13, "ራ": 14, ")": 15

}

Each word is then represented as a sequence of integers corresponding to the indices of its characters. For the word "ጉ.ተ", this would be:

encoded\_word = [1, 2, 3]

For the full sentence, we would represent each word as a sequence of integers:

encoded\_sentence = [

[1, 2, 3], # "ጉ.ተ"

[5, 6, 3, 7, 8, 9], # "ኢንተርኔት"

[10, 11], # "ካፌ"

[12, 1, 13, 3, 14, 14, 15] # "ጉና ተራራ"

]

### FastText-level Encoding

FastText-level encoding uses pre-trained word embedding to represent words as dense vectors. This embedding is learned based on the context in which words appear in large corpora and capture both syntactic and semantic information about the words. FastText, specifically, improves on traditional word embeddings by also considering sub-word information, which allows it to handle misspellings or variations in the form of a word, such as different tenses or derivations. For the sentence "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)", we use a pre-trained FastText model to generate dense vector embeddings for each word. The FastText model captures semantic and syntactic features for each word, representing them as high-dimensional vectors.

**Pseudocode:**

Load pre-trained FastText model

For each word in dataset:

Generate embedding using FastText model

Store embeddings for each word

**For example:**

fasttext\_model = load\_pretrained\_fasttext\_model()

embeddings = {

"ጉ.ተ": fasttext\_model.get\_word\_vector("ጉ.ተ"),

"ኢንተርኔት": fasttext\_model.get\_word\_vector("ኢንተርኔት"),

"ካፌ": fasttext\_model.get\_word\_vector("ካፌ"),

"ጉና": fasttext\_model.get\_word\_vector("ጉና"),

"ተራራ": fasttext\_model.get\_word\_vector("ተራራ")

}

Now, the sentence can be represented by a sequence of dense vectors corresponding to each word:

encoded\_sentence = [embeddings["ጉ.ተ"], embeddings["ኢንተርኔት"], embeddings["ካፌ"], embeddings["ጉና"], embeddings["ተራራ"] ]

### One-hot Encoding

One-hot encoding is one of the simplest and oldest encoding techniques. It represents each word or character as a binary vector. In this vector, the index corresponding to the word or character is set to 1, and all other indices are set to 0. Essentially, each word is represented as a vector with only one "hot" (active) value, indicating its position in the vocabulary, and all other positions are "cold" (inactive). For the sentence "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)", we first construct a vocabulary, where each word is assigned a unique index.

**Pseudocode:**

Initialize vocabulary = {}

For each word in dataset:

If word not in vocabulary:

Assign a unique index to the word

Create a binary vector of length = size of vocabulary

Set the index corresponding to the word to 1

**For example:**

vocabulary = { "ጉ.ተ": 1, "ኢንተርኔት": 2, "ካፌ": 3, "ጉና": 4, "ተራራ": 5 }

Each word is then represented as a binary vector, with the index corresponding to the word set to 1:

encoded\_sentence = [ [1, 0, 0, 0, 0], # "ጉ.ተ" [0, 1, 0, 0, 0], # "ኢንተርኔት" [0, 0, 1, 0, 0], # "ካፌ" [0, 0, 0, 1, 0], # "ጉና" [0, 0, 0, 0, 1] # "ተራራ" ]

### Padding

Padding is not an encoding technique per se, but rather a preprocessing step used in conjunction with other encoding methods. Padding ensures that all sequences (such as sentences or documents) are of uniform length. Many machine learning models, particularly neural networks, require input sequences to have consistent lengths. If a sequence is shorter than the desired length, padding adds zeros (or a specified padding value) to the end of the sequence to make it the correct length. For example, if a sentence is represented as the sequence [1, 2, 3, 4, 5] but the model requires a sequence of length 6, we would add padding (e.g., zero) at the end:

padded\_sentence = [1, 2, 3, 4, 5, 0] # 0 for padding

Padding ensures that all sequences have a consistent length, enabling the model to process them uniformly.

## Embedding Techniques

## Word Embedding

Pre-trained word embedding, such as Word2Vec or GloVe, are used to map words into continuous vector spaces where similar words have similar vectors. These embeddings are integrated into the system by loading pre-trained models and applying them to encode the tokenized trade names, enabling the machine learning models to leverage semantic relationships learned from large corpora. This integration enhances the model's ability to understand semantic meaning and context, ultimately improving the system’s performance in identifying trade names effectively.

### Pseudocode for Integrating Pre-trained Word Embedding

Load pre-trained word embedding model (e.g., Word2Vec or GloVe)

For each word in the tokenized dataset:

If word exists in the pre-trained model:

Retrieve the corresponding embedding vector

Else:

Assign a random vector of the same dimension

Store all word embeddings for input to machine learning models

### Example

**Input:** "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)"

**Output:**

[0.25, -0.15, 0.75, 0.40, -0.10] (vector for "ጉ.ተ")

[0.45, 0.20, -0.05, 0.60, 0.30] (vector for "ኢንተርኔት")

[0.35, 0.10, 0.50, -0.20, 0.55] (vector for "ካፌ")

[0.40, -0.05, 0.30, 0.45, 0.20] (vector for "(ጉና ተራራ)")

## Character Embedding

Character embedding captures subword-level information, enabling the model to learn from shared character sequences across different words. This is particularly useful for handling out-of-vocabulary words and spelling variations.

### Pseudocode for Generating Character Embedding

Initialize char\_set = {}

For each word in dataset:

For each character in word:

If character not in char\_set:

Assign a unique index to the character

Initialize embeddings\_matrix = Random values of shape (|char\_set|, embedding\_dim)

For each word in dataset:

Initialize word\_embedding = Zero vector of length embedding\_dim

For each character in word:

Get character index from char\_set

Add corresponding embedding vector from embeddings\_matrix to word\_embedding

Normalize word\_embedding

Store word\_embedding for each word

### Example

**Input:** "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)"

**Output:**

[0.12, 0.08, -0.05, 0.30, 0.15] (character-level embedding for "ጉ")

[0.25, -0.10, 0.20, 0.45, 0.35] (character-level embedding for "ተ")

...

## FastText Embeddings

FastText embeddings generate word representations by considering subword units (n-grams), enabling the model to handle morphological variations effectively. This approach allows the model to create embeddings for unseen words by combining subword representations.

### Pseudocode for Applying FastText Embeddings

Load pre-trained FastText model

For each word in the dataset:

Split the word into subword units (n-grams)

Retrieve or compute subword embeddings using FastText model

Combine subword embeddings to form the word embedding (e.g., by averaging)

Store the resulting word embeddings for use in downstream tasks

### Example

**Input:** "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)"

**Output:**

[0.30, 0.10, 0.25, 0.15, 0.40] (FastText embedding for "ጉ.ተ")

[0.50, 0.35, 0.20, -0.10, 0.55] (FastText embedding for "ኢንተርኔት")

...

## Concatenated Embedding

Concatenated embedding enhances model performance by combining diverse types of information from multiple embedding methods. Each embedding captures a unique aspect of the data: word-level embedding encodes semantic context, character-level embedding captures subword information, and FastText embedding handles morphological variations. By concatenating these embeddings, the model gains a richer, more comprehensive representation of trade names, which improves its ability to generalize across different scenarios.

### Pseudocode for Concatenating Embedding

Initialize embedding = []

For each word in dataset:

Get word embedding

Get character embedding

Get FastText embedding

Concatenate all embeddings

Append concatenated embedding to embedding list

### Mathematical Representation

Let:

* **W** represent the word-level embedding vector.
* **C** represent the character-level embedding vector.
* **F** represent the FastText embedding vector.

The concatenated embedding vector **E** is given by:

E=[W;C;F]E = [W; C; F]

### Example

**Input:** "ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)"

**Output:**

[0.25, -0.15, 0.75, 0.40, -0.10, 0.12, 0.08, -0.05, 0.30, 0.15, 0.30, 0.10, 0.25, 0.15, 0.40]

(Concatenated embedding combining word, character, and FastText embeddings)

This combined representation allows the model to leverage complementary information from different encoding techniques, resulting in improved detection and validation of trade names. The embedding generated from different encoding techniques are concatenated to form a comprehensive representation of each trade name. These concatenated embedding are then used in downstream tasks such as classification and similarity detection. By combining information from word, character, and FastText embedding, the model benefits from diverse data representations, which improve its ability to generalize and accurately identify trade names across varying contexts.

## Machine Learning Process

## Trade Name Compliance Detection Model

This model is trained using the embedding to identify whether a trade name complies with regulatory standards. The training phase is carried out in the **Train Compliance Detection Model** component, which results in the **Trade Name Compliance Detection Model**.

# Prohibited Names

To protect public interests and ensure clarity in business identification, certain categories of trade names are prohibited under regulatory guidelines. The **Trade Name Compliance Detection Model** automatically flags such names, preventing the registration of names that could offend, or breach ethical and legal standards.

# Government-Related Names

Trade names implying an association with government bodies or public institutions are prohibited. This restriction prevents potential confusion about a business's legitimacy or endorsement by a government entity.

**Examples:**

* **"ኮሚሽን"** (Commission)
* **"ኤጀንሲ"** (Agency)

Names that convey official status or authority, whether directly or indirectly, are not allowed.

# Names of Political Parties, Trade Unions, or Charitable Organizations

Names identical to or resembling those of political entities, labor unions, or charitable institutions are banned. This ensures that business names are not used to falsely imply affiliation with well-known organizations or movements.

**Example:**  
A name like **"ፋኦ የወተት ልማት ንግድ"** would be disqualified if it conflicts with **FAO**, a globally recognized entity (Food and Agriculture Organization).

# Famous Personal Names

The use of famous personal names in trade names is restricted to prevent misuse and to safeguard the reputation of prominent individuals. Using a well-known name without authorization may imply false representation or endorsement.

**Example:**

* **"ሀይሌ ገብረስላሴ"** – The name of a world-renowned athlete cannot be registered without explicit consent.

# Globally Recognized Brand Names

Trade names identical to or closely resembling internationally established brands or organizations are prohibited unless prior approval is obtained from the brand owner. This prevents trademark infringement and deceptive practices.

**Examples:**

* **"ቶዮታ"** (Toyota)
* **"ማይክሮሶፍት"** (Microsoft)

Such names are protected under international intellectual property laws, and unauthorized registration is not allowed.

# Offensive or Inappropriate Names

Trade names containing offensive, derogatory, or inappropriate language are strictly banned. This includes names that insult or degrade individuals, communities, or institutions. The model detects such names to maintain professional and ethical standards in business environments.

**Categories:**

* **Insulting religion** – Names that disrespect or offend religious beliefs or symbols.
* **Insulting language, nationality, or ethnicity** – Names with offensive language targeting specific groups.
* **Promoting immoral or illegal behavior** – Names that glorify drunkenness, promiscuity, drug use, crime, or other unethical practices.
* **Defaming reputation** – Names that harm the reputation of individuals or organizations.
* **Subordinate gender terms** – Names that imply gender inequality or discrimination.
* **Encouraging war or violence** – Names that promote conflict or violence against others.
* **Incitement to hate** – Names that encourage hatred or hostility based on race, religion, nationality, or gender.
* **Damaging national reputation** – Names that tarnish the image or dignity of a nation or its people.

## Trade Name Similarity Ambiguity Detection Model

The **Trade Name Similarity Ambiguity Detection Model** is designed to evaluate the degree of similarity between trade names and identify potential ambiguities that may arise when registering new business names. This model helps mitigate legal, operational, and branding risks by ensuring that newly proposed business names are unique, non-misleading, and distinct from pre-registered names.

Incorporating advanced machine learning techniques, the model utilizes embeddings during the **Training Phase** to create a robust mechanism for detecting ambiguous names. This ensures the integrity of trade name registration processes and fosters transparency in business environments. Below are several key guidelines, supported by the model, which help users avoid ambiguity and ensure compliance when selecting new trade names.

# Adding Adjectives or Similar Words

Adding minor variations or adjectives to a registered trade name is often insufficient to make the new name distinct. The model flags such names as potentially misleading, preventing the registration of names that closely resemble existing ones. For instance, if "ሕብረት ቡና ንግድ" is already registered, the model will bring about the following names would not be acceptable:

* የሕብረት ቡና ንግድ
* ሕብረት ቡና ንግድቁ. 2
* ሕብረት የቡና ንግድ

However, adding completely new, unrelated words can make a name distinct and acceptable.

For example:

* እድገት በሕብረት ቡና ንግድ
* ሕብረት ለስራ ቡና ንግድ

# Changing the Order of Words

Simply rearranging the order of words in an existing business name does not always result in a distinct name. The model recognizes that such rearrangements can still cause confusion, and therefore, it flags names that might be misleading.

**Example:**   
If "ሕብረት የቡናና ሻይ ንግድ" is already registered, then "ሕብረት የሻይና ቡና ንግድ" would be flagged as misleading.

# Using Personal Names in Business Names

The use of personal names in business names is permissible, provided they do not duplicate or closely resemble pre-existing trade names. The model ensures that even if personal names are involved, ambiguity and confusion are minimized.

**Example:**   
If "ወንድወሰን የኤሌክትሪክ ዕቃዎች ንግድ" is registered, similar names must include significant distinctions:

* "ወንድወሰን ቶላ ጉርሜ የኤሌክትሪክ ዕቃዎች ንግድ"
* "ወንድወሰን 8798 የኤሌክትሪክ ዕቃዎች ንግድ"

Additionally, gender-specific variations in names are recognized as distinct by the model.

**Examples:**

* "ታደለ"vs**.** "ታደለች"
* "አበበ"vs**.** "አበበች"

# Pronunciation Similarity

Even if trade names are spelled differently, similar pronunciation can create ambiguity. The model detects and flags names with phonetic similarities to prevent misleading registrations.

**Examples:**

* "ኩልል የመጠጥ ውሃ" vs. "ኩል የመጠጥ ውሃ"
* "Kool water" vs**.** "Cool water"

# Meaning Similarity

Names with different wording but similar meanings are typically flagged by the model only when the similarity may cause confusion. However, names with clear contextual differences may still be acceptable.

**Examples:**

* "አስተማማኝ ኮንስትራክሽን" and "ሪሊይብል ኮንስትራክሽን"are acceptable despite their similar meanings because they are contextually distinct.

# Types of Business Represented

Trade names representing similar or related types of businesses are carefully reviewed by the model to avoid ambiguity.

**Example:**  
If **"ሞሃ የለስላሳ መጠጦች ኢንደስትሪ"** is registered, then:

* **"ሞሃ የወፍጮ ንግድ ሥራ"**
* **"ሞሃ ባልትና"**  
  would be considered distinct because they represent different business types.

# Acronyms in Business Names

The model assures acronyms to be acceptable as trade names if they represent specific, meaningful terms. The model evaluates the uniqueness and significance of the acronym before approval.

**Examples:**

* **"ጉ.ተ ኢንተርኔት ካፌ (ጉና ተራራ)"**
* **"አ.ም ልጅ (አርባ ምንጭ)"**
* **"መ.ታ ኮንሰልተንሲ (መልቲ - ታሇንት)"**

Here’s the essay formatted in a document style for your convenience:

## Using CNN, RNN, and GPT for Trade Name Compliance and Similarity Ambiguity Detection Models

In natural language processing (NLP) and machine learning, detecting compliance with trade name regulations and identifying trade name similarity ambiguity are critical tasks. To address these challenges, different deep learning models can be employed to extract features from text data and make predictions. Among the most prominent models for such tasks are **Convolutional Neural Networks (CNNs)**, **Recurrent Neural Networks (RNNs)**, and **Generative Pretrained Transformers (GPTs)**. These models have been successfully applied to a wide range of NLP problems, including text classification, sequence modeling, and contextual understanding. In this essay, we explore how these models can be used for the **Trade Name Compliance Detection Model** and **Trade Name Similarity Ambiguity Detection Model**, with detailed mathematical descriptions of their operations.

## Trade Name Compliance Detection Model Using CNN

# CNN for Feature Extraction

Convolutional Neural Networks (CNNs) are particularly effective for extracting local features from data, especially when the data has a spatial or sequential structure. In the context of trade name compliance detection, CNNs can be applied to **word-level embeddings** or **character-level embeddings** to identify important patterns and features that indicate compliance or non-compliance.

1. **Input Representation**:  
   The input to the CNN can be a sequence of word embeddings representing the trade name. Let’s denote the trade name as a sequence of words:

X=[x1,x2,…,xn]X = [x\_1, x\_2, \dots, x\_n]

where xix\_i represents the embedding of the ii-th word in the trade name.

1. **Convolution Layer**:  
   The CNN applies a set of convolutional filters WkW\_k (with k=1,2,…,Kk = 1, 2, \dots, K) to the input sequence. The convolution operation computes feature maps by sliding the filters across the sequence:

hk=ReLU(Wk⋅X+bk)h\_k = \text{ReLU}(W\_k \cdot X + b\_k)

where bkb\_k is the bias term and hkh\_k represents the feature map produced by filter WkW\_k.

1. **Pooling**:  
   A **max-pooling** operation is typically applied to each feature map to reduce dimensionality and highlight the most important features:

pk=max⁡(hk)p\_k = \max(h\_k)

where pkp\_k is the pooled feature for filter kk.

1. **Fully Connected Layer**:  
   The pooled features from all filters are concatenated and passed through a fully connected layer, followed by a softmax function to predict the compliance label:

y^=Softmax(Wfc⋅p+bfc)\hat{y} = \text{Softmax}(W\_{\text{fc}} \cdot p + b\_{\text{fc}})

where WfcW\_{\text{fc}} and bfcb\_{\text{fc}} are the weight matrix and bias for the fully connected layer.

## Trade Name Compliance Detection Model Using RNN

# RNN for Sequential Dependencies

Recurrent Neural Networks (RNNs) are designed to capture the sequential dependencies in data, making them ideal for tasks where the order of elements matters, such as analyzing a trade name’s structure. RNNs can process the trade name word-by-word and learn the temporal dependencies between words.

1. **Input Representation**:  
   The input to the RNN is a sequence of word embeddings, similar to the CNN model:

X=[x1,x2,…,xn]X = [x\_1, x\_2, \dots, x\_n]

1. **RNN Recurrence**:  
   The RNN processes the sequence of word embeddings one time step at a time. At each time step tt, the hidden state hth\_t is updated using the previous hidden state ht−1h\_{t-1} and the current input xtx\_t:

ht=tanh(Wrnn⋅[ht−1,xt]+brnn)h\_t = \text{tanh}(W\_{\text{rnn}} \cdot [h\_{t-1}, x\_t] + b\_{\text{rnn}})

1. **Prediction**:  
   After processing all words in the trade name, the final hidden state hnh\_n is passed through a softmax layer to predict whether the trade name is compliant or non-compliant:

y^=Softmax(Wfc⋅hn+bfc)\hat{y} = \text{Softmax}(W\_{\text{fc}} \cdot h\_n + b\_{\text{fc}})

## Trade Name Compliance Detection Model Using GPT

# GPT for Contextual Understanding

Generative Pretrained Transformers (GPTs) have revolutionized NLP by using attention mechanisms to capture contextual dependencies between words in a sequence. GPT can be applied to trade name compliance detection by providing a **contextualized understanding** of the trade name, considering both its internal structure and its surrounding context.

1. **Input Representation**:  
   The trade name is tokenized and passed as input to the GPT model. Each token tit\_i is represented as an embedding:

ti=Embedding(wi)t\_i = \text{Embedding}(w\_i)

where wiw\_i represents the ii-th word in the trade name.

1. **Self-Attention Mechanism**:  
   GPT uses a multi-head self-attention mechanism to model the relationships between all words in the sequence. The attention mechanism computes the attention weights AA for each word in relation to all other words:

Aij=exp⁡(qi⋅kj)∑jexp⁡(qi⋅kj)A\_{ij} = \frac{\exp(q\_i \cdot k\_j)}{\sum\_{j} \exp(q\_i \cdot k\_j)}

where qiq\_i and kjk\_j are the query and key vectors for the ii-th and jj-th words, respectively.

1. **Contextualized Output**:  
   The final hidden states are passed through a linear layer and a softmax function to predict compliance:

y^=Softmax(Wfc⋅hn+bfc)\hat{y} = \text{Softmax}(W\_{\text{fc}} \cdot h\_n + b\_{\text{fc}})

## Using CNN, RNN, and GPT for Trade Name Similarity Ambiguity Detection Model

A **Trade Name Similarity Ambiguity Detection Model** aims to detect when two trade names are semantically or syntactically similar, potentially causing confusion or ambiguity.

## Trade Name Similarity Ambiguity Detection Model Using CNN

# CNN for Similarity Detection

CNNs can be used to capture local patterns between pairs of trade names by applying convolutional filters to their embeddings. The model can learn distinguishing features that highlight similarities or differences between two trade names.

1. **Input Representation**:  
   Two trade names X1=[x1,x2,…,xn]X\_1 = [x\_1, x\_2, \dots, x\_n] and X2=[y1,y2,…,ym]X\_2 = [y\_1, y\_2, \dots, y\_m] are encoded into sequences of word embeddings.
2. **Convolutional Filters**:  
   CNNs apply filters to both sequences to capture local patterns:

h1=ReLU(Wk⋅X1+bk),h2=ReLU(Wk⋅X2+bk)h\_1 = \text{ReLU}(W\_k \cdot X\_1 + b\_k), \quad h\_2 = \text{ReLU}(W\_k \cdot X\_2 + b\_k)

1. **Similarity Prediction**:  
   After pooling, the similarity between the two sequences is calculated using a similarity metric such as cosine similarity:

similarity=h1⋅h2∥h1∥∥h2∥\text{similarity} = \frac{h\_1 \cdot h\_2}{\|h\_1\| \|h\_2\|}

The model then predicts whether the trade names are similar or not based on this similarity score.

## Trade Name Similarity Ambiguity Detection Model Using RNN

# RNN for Sequential Similarity

RNNs can be applied to capture the sequential dependencies between the words of two trade names, considering the context of each word in both names.

1. **Input Representation**:  
   The two trade names are processed in parallel by two RNNs. At each time step, the hidden state is updated for both sequences:

h1,t=tanh(Wrnn⋅[h1,t−1,xt]+brnn)h\_{1,t} = \text{tanh}(W\_{\text{rnn}} \cdot [h\_{1,t-1}, x\_t] + b\_{\text{rnn}}) h2,t=tanh(Wrnn⋅[h2,t−1,yt]+brnn)h\_{2,t} = \text{tanh}(W\_{\text{rnn}} \cdot [h\_{2,t-1}, y\_t] + b\_{\text{rnn}})

1. **Similarity Computation**:  
   After processing both sequences, the final hidden states h1,nh\_{1,n} and h2,mh\_{2,m} are compared using a similarity metric:

similarity=h1,n⋅h2,m∥h1,n∥∥h2,m∥\text{similarity} = \frac{h\_{1,n} \cdot h\_{2,m}}{\|h\_{1,n}\| \|h\_{2,m}\|}

## Trade Name Similarity Ambiguity Detection Model Using GPT

# GPT for Contextualized Similarity

GPT can be used to capture deeper, contextual relationships between trade names, utilizing its attention mechanism to compute pairwise similarity.

1. **Input Representation**:  
   Both trade names are tokenized and passed into the GPT model, which computes contextual embeddings for each word in both sequences.
2. **Similarity Prediction**:  
   The similarity score is derived from the attention weights, reflecting how the model perceives the relationship between the two trade names.

CNNs, RNNs, and GPTs are powerful models that can be effectively utilized for **Trade Name Compliance Detection** and **Trade Name Similarity Ambiguity Detection**. While CNNs excel at capturing local patterns, RNNs model sequential dependencies, and GPTs provide contextualized embeddings, together, these models offer a comprehensive approach to detecting compliance and similarity. The combination of these techniques allows for more accurate and efficient detection, enabling businesses to better manage trade name disputes and regulatory compliance.

## Training

The training process for a combined CNN, RNN, and GPT model begins with the initialization of the architecture, where each component serves a unique role in capturing various aspects of the data. The **Convolutional Neural Network (CNN)** is designed to capture local features from the input text, such as word patterns, character-level information, and other important cues. The CNN operates over fixed-size windows of the input and generates feature maps, which highlight the key information within the input sequences. This process is critical for extracting fundamental patterns, particularly useful when dealing with noisy text or word-level irregularities. The **Recurrent Neural Network (RNN)**, on the other hand, is responsible for learning the sequential dependencies between words, phrases, and sentences. It processes the data in a time-dependent manner, maintaining a memory of past inputs through hidden states, which helps in modeling the relationships between different words across the sequence. The **Generative Pretrained Transformer (GPT)** component works on a more global level, capturing long-range contextual relationships within the input text. By leveraging a self-attention mechanism, the GPT model allows the model to focus on different parts of the sequence simultaneously, understanding how each word relates to the rest of the text, regardless of their position. After each of these components processes the data, their outputs are concatenated into a unified representation, which is passed to the final decision layer. This decision layer generates the output prediction, which could be a classification label for detecting compliance or a similarity score for resolving ambiguities in trade names. The fusion of CNN, RNN, and GPT allows the model to leverage the strengths of each architecture, providing a comprehensive understanding of the input text.

## Optimization

The **optimization process** is a critical phase in training the model, aiming to minimize the loss function and improve the model’s ability to make accurate predictions. To begin with, after the model makes a prediction, the **loss function** measures the difference between the predicted output and the true label. For classification tasks, this is often done using **cross-entropy loss** [85], which quantifies how well the model’s predicted probabilities match the true labels. The goal of the optimization process is to adjust the model's parameters (weights) so that the loss is minimized. **Backpropagation** [86] is then employed, where the gradients of the loss with respect to each weight are calculated. This step is crucial because it informs the optimization algorithm of how much each weight contributed to the error, enabling the network to adjust accordingly. Once the gradients are computed, the model employs **gradient descent** [87] to update its weights iteratively. This involves taking small steps in the direction of the negative gradient, which helps reduce the loss over time. One of the key factors in gradient descent is the **learning rate**, which controls the size of these steps. A learning rate that is too large may cause the model to miss the optimal weights, while a rate that is too small could lead to slow or incomplete convergence. To accelerate and stabilize this process, advanced optimizers like **Adam (Adaptive Moment Estimation)** [88] or **RMSProp** [89] are often used. These optimizers adjust the learning rate for each parameter individually, making the training process more efficient and ensuring faster convergence, especially when dealing with sparse or noisy gradients.

Regularization is also a vital part of the optimization process. Without it, the model is at risk of overfitting, where it memorizes the training data rather than generalizing to new, unseen data. Regularization techniques, such as **L2 regularization (Ridge)** [90], are incorporated into the loss function. This penalty term discourages the model from learning overly large weights, which can lead to overfitting. The L2 penalty term is proportional to the sum of the squares of the weights and is added to the original loss function. Another common technique is **dropout** [91], where a random subset of neurons is deactivated during training, forcing the model to learn more robust features by preventing it from becoming overly reliant on any single neuron. **Early stopping** [92] is another regularization technique that monitors the model’s performance on the validation set. If performance starts to degrade, it halts training before the model begins to overfit to the training data. Regularization methods like these help ensure that the model generalizes well to unseen data, enhancing its performance on real-world tasks.

The final part of the optimization process involves **hyperparameter tuning** [93], where crucial settings such as the learning rate, batch size, the number of layers in the model, and the overall architecture are adjusted to maximize performance. Hyperparameters are typically set before training begins, but their optimal values often need to be found through experimentation. Methods such as **grid search** [94], which tests a wide range of hyperparameter combinations, or **random search** [95], which samples hyperparameters randomly from a predefined space, are often used to explore the most effective configurations. More advanced techniques, like **Bayesian optimization** [96], can further streamline this process by using probabilistic models to predict the best-performing hyperparameters based on previous training runs. Proper tuning of these hyperparameters is essential for achieving the best performance and ensuring the model converges to an optimal solution without unnecessary computational costs.

In summary, the combined CNN, RNN, and GPT model is trained and optimized through a series of well-coordinated steps. By integrating these different model architectures, the model is able to capture local, sequential, and contextual features, making it well-suited for tasks like **Trade Name Compliance Detection** and **Trade Name Similarity Ambiguity Detection**. The optimization process, which includes gradient descent, advanced optimizers, regularization techniques, and hyperparameter tuning, plays a crucial role in improving the model’s accuracy and generalization ability. These processes ensure that the model can learn effectively from the training data and provide accurate predictions when applied to real-world scenarios.

## Prediction

The **Trade Name Compliance Detection Model** aims to determine if a proposed trade name complies with certain regulations, rules, or predefined criteria. The prediction process involves several key steps. First, in the input preprocessing step, the proposed trade name, typically in the form of raw text, undergoes preprocessing, including tasks like tokenization, normalization (such as converting to lowercase), and handling any special characters. After this, the preprocessed text is fed into the trained model. In the feature extraction step, the trade name is passed through the model, which may involve components like a Convolutional Neural Network (CNN) to capture local features and Recurrent Neural Networks (RNNs), such as LSTMs, to understand the sequential relationships between words within the trade name. Additionally, a Generative Pretrained Transformer (GPT) might process contextual information and long-range dependencies. The combined feature representations from CNN, RNN, and GPT are then passed through a decision layer that classifies the trade name into one of the predefined classes, such as "Compliant" or "Non-Compliant" based on the learned criteria. The model computes a probability score for each class, typically using a softmax function. Finally, the output of the model is a predicted class label, such as "Compliant" (if the trade name adheres to the compliance rules) or "Non-Compliant" (if the trade name violates one or more compliance rules). If necessary, the output can include the probability of the trade name belonging to each class, offering additional insight into the confidence of the prediction.

On the other hand, the **Trade Name Similarity Ambiguity Detection Model** focuses on identifying whether a proposed trade name is similar or ambiguous when compared to existing trade names, potentially leading to confusion or legal challenges. This process also involves several steps. First, as with the compliance model, the input trade name is preprocessed, which involves tokenizing and normalizing the trade name. In addition to this, a comparison trade name (or multiple trade names) might also be provided to check for similarity. In the feature representation and comparison step, the preprocessed trade names are passed through the combined CNN, RNN, and GPT layers. These layers help generate both local feature maps (from CNN), sequential relationships (from RNN), and contextual embeddings (from GPT). To compute similarity, a cosine similarity measure or other distance metrics can be employed to compare the output embeddings of the proposed trade name with those of existing trade names in the model’s database. In the ambiguity detection step, the model evaluates how similar the proposed trade name is to known trade names using the similarity measures. If the trade name is very close to existing trade names, the model may output a label indicating ambiguity. This label could be "Ambiguous" (if the trade name is too similar to existing ones and could lead to confusion) or "Unique" (if the trade name does not have significant similarity to other trade names). Finally, the model’s output is the prediction of whether the trade name is ambiguous or unique, based on its similarity to existing trade names. The model could also output a similarity score, such as a cosine similarity value between 0 and 1, to indicate how close the proposed trade name is to the most similar trade name in the database.

# Chapter Five: Experimentation and Results

## Introduction

This chapter presents the experimental setup, implementation details, evaluation methodology, and results of the proposed **name compliance detection models** and **Trade name similarity detection models.**

**Part one of this chapter presents the experimentation and result of the Trade name compliance detection models,** whichmultiple deep learning architectures were implemented and evaluated to assess their effectiveness in automatically determining whether a trade name complies with registration rules. The experiments were conducted using word-level, character-level, combined representations, and FastText-based embeddings integrated with Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Part two of this chapter presents the experimentation and results of the **trade name similarity detection model**, which is designed to identify newly proposed trade names that are identical or highly similar to already registered trade names. The similarity detection model complements the trade name compliance detection model by addressing duplication, semantic resemblance, and phonetic similarity that may lead to confusion in the registration process. Multiple text representation and embedding techniques were implemented and evaluated, including TF-IDF, FastText, BERT token embeddings, and Sentence-BERT. The performance of these models was analyzed under a unified experimental setup to assess their effectiveness in detecting similar trade names.

## The Dataset

The trade name dataset was collected from various sources those sources include tomerakato.com. Amharic sentences for the purpose of compliance detection model were collected from various sources, these sources include the Amharic Bible, Addis Zemen Gazette, The Ethiopian Reporter Newspaper, Ethiopian Federal Negarit Gazette, and from corpus collected for Amharic-English machine translation8. Those collected data were converted to structured CSV file containing two main attributes:

* **trade\_name**: textual representation of the proposed trade name
* **reason**: categorical label representing the compliance decision or rejection reason

The dataset was preprocessed by:

* Removing missing values
* Normalizing whitespace
* Converting all proposed trade names to string format

The labels were encoded using **Label Encoding** and transformed into **one-hot vectors** for multi-class classification. The dataset was split automatically during training using a **validation split of 15%**, ensuring consistent evaluation across all models.

The dataset used for the similarity detection experiment consists of **Amharic trade names**, labeled with their registration status (registered or unregistered). The registered trade names form the reference database against which newly submitted trade names are compared.

Each record in the dataset contains:

* Trade name text
* Registration status
* Binary label derived from registration status

The dataset was preprocessed by converting all trade names into string format. No aggressive text normalization was applied to preserve linguistic characteristics essential for similarity computation.

## Experimentation Environment

## Trade name compliance detection models

### Host Computer

The experiments were conducted in a cloud-based environment with GPU acceleration, enabling efficient training of deep learning models. Persistent storage was used to save trained models, tokenizers, logs, and checkpoints for reproducibility and resumption of training.

### Development Tools

We used a combination of well-established tools and libraries to support data processing, model development, and deployment. **Python** served as the core programming language for implementing the entire workflow due to its rich ecosystem and ease of integration with machine learning libraries. **TensorFlow with Keras** was used for designing, training, and evaluating deep learning models; specifically, the Keras *functional API* was adopted to enable flexible and scalable multi-input neural network architectures. **Gensim** was utilized to train **FastText** word embeddings, allowing the models to capture semantic and subword information from textual data. For preprocessing tasks such as label encoding, **Scikit-learn** was employed due to its reliability and simplicity. **NumPy** and **Pandas** were used extensively for numerical computation, data manipulation, and efficient handling of structured datasets. Finally, **Gradio** was integrated to provide an interactive interface for model inference, enabling real-time testing and demonstration of the trained models.

<https://www.python.org>

<https://www.tensorflow.org>

<https://scikit-learn.org><https://numpy.org>

<https://www.gradio.app>

### Building the Model

We investigate ten distinct deep learning models organized into five conceptual families, each designed to capture different linguistic properties of trade names. This structured grouping allows a systematic comparison of representation strategies and modeling assumptions across word-level, character-level, and hybrid approaches.

**Word-level models**, namely *cnn\_word* and *rnn\_word*, operate on tokenized sequences of words with a fixed maximum length. In these architectures, word embeddings are learned from scratch during training, allowing the models to capture semantic regularities and contextual patterns present in the dataset. Convolutional neural networks (CNNs) focus on extracting local n-gram–like features, while recurrent neural networks (RNNs) model sequential dependencies across words. However, despite their ability to encode semantic meaning, word-level models are inherently sensitive to out-of-vocabulary (OOV) issues. This limitation is particularly pronounced in trade names, which often contain invented terms, uncommon spellings, or domain-specific vocabulary not seen during training.

**Character-level models**, represented by *cnn\_char* and *rnn\_char*, address these limitations by operating directly on sequences of characters rather than words. Character sequences are explicitly constructed using a custom vocabulary that includes special tokens such as <PAD> for sequence alignment and <OOV> for unseen characters. By modeling text at the character level, these architectures become robust to spelling variations, transliteration differences, abbreviations, and creative morphological constructions. Such properties are common in trade names, making character-level representations especially effective in capturing orthographic patterns that word-level models may fail to recognize.

**Combined word and character models**, namely *cnn\_combined* and *rnn\_combined*, integrate the strengths of both representations through dual-branch architecture. One branch processes word-level inputs to learn semantic information, while the other processes character-level inputs to capture orthographic and subword patterns. The feature representations learned by these two branches are concatenated before the final classification layer. This fusion strategy enables the model to jointly reason about both meaning and surface form, thereby mitigating the weaknesses of using either word-level or character-level representations in isolation.

**FastText (Keras-style) hybrid models**, consisting of *cnn\_fasttext\_keras* and *rnn\_fasttext\_keras*, introduce higher-dimensional word embeddings inspired by FastText but implemented using standard Keras embedding layers. In this setting, embeddings are trained end-to-end during model training without explicit subword or character n-gram modeling. These models serve as a controlled baseline, allowing the experiment to isolate the impact of embedding dimensionality and neural architecture from the benefits introduced by true subword-aware embeddings.

Finally, **FastText (Gensim) hybrid models**, namely *cnn\_fasttext\_gensim* and *rnn\_fasttext\_gensim*, incorporate true FastText embeddings trained using the Gensim library. These embeddings explicitly model character n-grams and subword information, enabling the representation of rare, unseen, or morphologically complex words. The precomputed FastText vectors are supplied directly to the neural network as input embeddings. This approach is particularly well suited for trade name analysis, where texts are typically short, noisy, and dominated by rare or invented terms, and where subword composition provides critical semantic and phonetic cues.

The architectural and training design of the models was carefully aligned with the linguistic characteristics of trade names and the experimental objectives, ensuring consistency across all model families.

**CNN-based models** employ multiple convolutional kernel sizes (2, 3, 4, and 5), followed by a global max-pooling layer. This design allows each kernel to capture local n-gram–like patterns of varying lengths, ranging from short character or word fragments to slightly longer lexical constructions. The use of global max pooling enforces position invariance, enabling the models to detect the most salient features regardless of where they occur within the trade name. As a result, CNNs are particularly effective at identifying distinctive lexical or orthographic cues that signal similarity or validity, even in short and irregular text sequences.

**RNN-based models** are implemented using bidirectional Long Short-Term Memory (BiLSTM) networks across all recurrent variants. By processing sequences in both forward and backward directions, BiLSTMs capture contextual information that depends on token ordering and local dependencies. This capability is useful when semantic interpretation is influenced by the arrangement of characters or words. However, because trade names are typically short, the primary benefit of RNNs in this setting lies in contextual refinement and sequence smoothing rather than modeling long-range dependencies.

To address overfitting risks, a **consistent regularization strategy** is applied across all architectures through the use of dropout layers. Given the limited length of input sequences and the potentially constrained size of the training dataset, dropout serves as a crucial mechanism to reduce co-adaptation of neurons, improve generalization, and stabilize model performance across different experimental runs.

The **training protocol** is standardized for all models to ensure fair comparison and reproducibility. Optimization is performed using the Adam optimizer, paired with categorical cross-entropy as the loss function, which is appropriate for multi-class classification tasks. Models are trained with a batch size of 64, and 15% of the training data is reserved as a validation set to monitor generalization performance during training.

To further enhance robustness and efficiency, **early stopping and checkpointing** mechanisms are employed. The best-performing model is selected based on minimum validation loss, preventing unnecessary overtraining once performance plateaus. In addition, training states are periodically saved, enabling epoch-level resumption in the event of interruption. This design choice ensures computational efficiency, experimental reproducibility, and protection against catastrophic forgetting during extended experimental runs.

#### Model Parameters and Training

All models were trained using the Adam optimizer in combination with the categorical cross-entropy loss function, a standard and effective choice for multi-class classification problems. To ensure a comprehensive assessment of model performance beyond accuracy, custom evaluation metrics were implemented to compute precision, recall, and F1-score during both training and validation phases. These metrics provide deeper insight into class-wise prediction quality and are particularly important in scenarios where class imbalance or misclassification costs may be significant.

The training process was carefully managed using multiple control mechanisms to improve robustness and reproducibility. The **ModelCheckpoint** callback was employed to automatically save the best-performing model based on validation performance, ensuring that the optimal model parameters were preserved. **EarlyStopping** was used to halt training when validation performance ceased to improve, thereby preventing overfitting and unnecessary computation. In addition, custom epoch-level tracking was implemented to allow interrupted training sessions to be resumed seamlessly, which is especially useful for long or resource-constrained experimental runs. Each model was trained for a maximum of five epochs with a batch size of 64, balancing training efficiency with convergence stability.

Model hyperparameters were standardized across experiments to enable fair comparison between architectures. The maximum word sequence length was set to 25 tokens, while character-level inputs were capped at 200 characters to capture sufficient orthographic detail without excessive padding. Word embeddings were learned with a dimensionality of 100, and character embeddings were set to 64 dimensions, providing a compact yet expressive representation space. For CNN-based architectures, convolutional kernel sizes of 2, 3, and 4 were used, with 128 filters allocated per kernel to ensure adequate feature extraction capacity. Recurrent models employed LSTM layers with 128 hidden units, and a dropout rate of 0.5 was applied across models to reduce overfitting. Training was conducted with a batch size of 64, a validation split of 15%, and a maximum of five epochs for all neural models.

For the FastText models trained using the Gensim library, a separate set of embedding-specific parameters was defined to fully exploit subword information. Embeddings were trained with a vector size of 300, a context window size of 5, and a skip-gram architecture, which is well suited for learning high-quality representations of rare and morphologically rich words. These embeddings were trained for five epochs before being integrated into the downstream neural network models.

| **Parameter** | **Value** |
| --- | --- |
| Maximum word length | 25 |
| Maximum character length | 200 |
| Word embedding dimension | 100 |
| Character embedding dimension | 64 |
| CNN filters | [2, 3, 4] |
| Number of CNN filters | 128 |
| LSTM units | 128 |
| Dropout rate | 0.5 |
| Batch size | 64 |
| Epochs | 5 |
| Validation split | 15% |

For FastText (Gensim-based):

* Vector size: 300
* Window size: 5
* Skip-gram architecture
* Five training epochs

### Evaluation

### Experimental Scenario

Each model was trained independently using the same dataset and configuration parameters to ensure a fair comparison. Performance was evaluated on the validation subset generated during training. The best model for each architecture was selected based on **minimum validation loss**.

### Evaluation Metrics

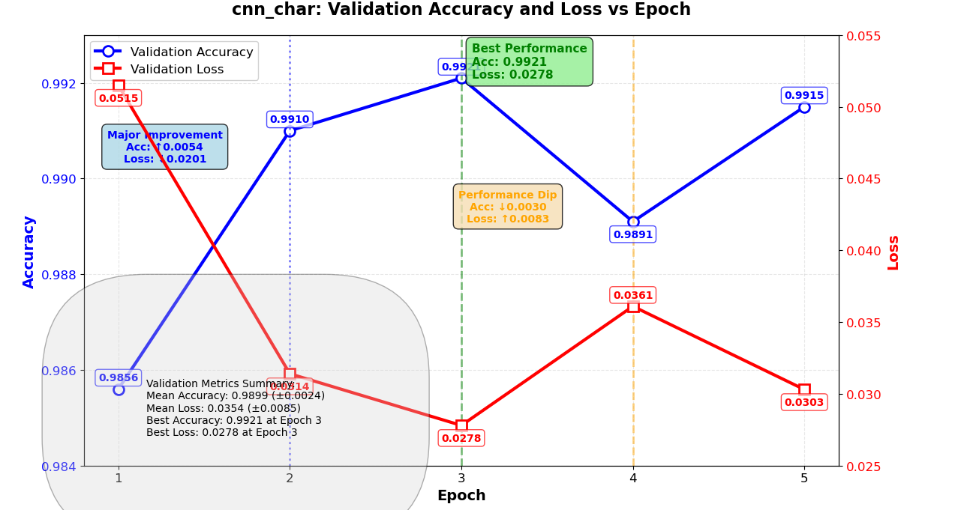
The following evaluation metrics were used:

* **Accuracy**: Overall correctness of predictions
* **Precision**: Proportion of correctly predicted compliant trade names
* **Recall**: Ability to correctly identify all relevant compliance cases
* **F1-Score**: Harmonic mean of precision and recall

These metrics were computed during training and validation using custom Keras backend functions.

### Test Results

The experimental results indicate that models utilizing **character-level and FastText-based representations** achieved superior performance compared to word-only models. In particular:

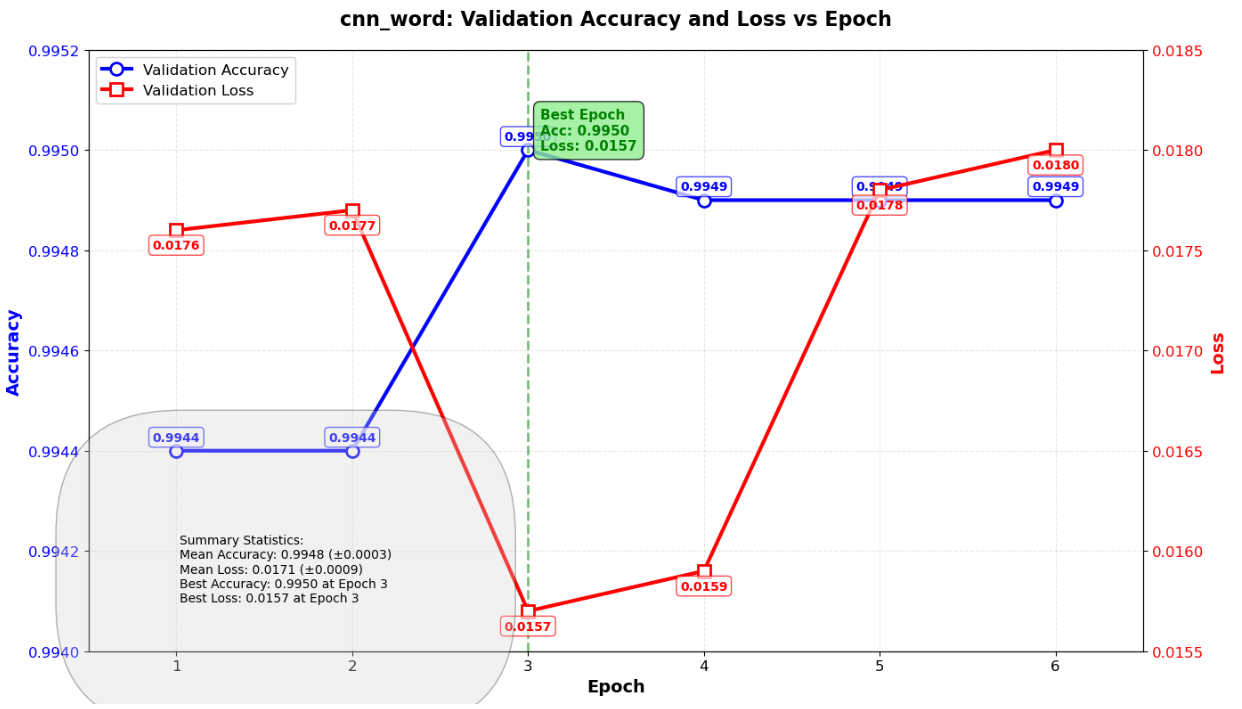
CNN Char: 

The CNN Char model demonstrated significant learning improvements throughout its training process, starting from a modest baseline and achieving substantial gains in performance. Beginning with a validation accuracy of 0.9856 and relatively high loss of 0.0515 at Epoch 1, the model exhibited remarkable improvement in Epoch 2, with accuracy increasing substantially by 0.0054 to 0.9910 and loss decreasing dramatically by 0.0201 to 0.0314. This early major improvement indicates effective feature learning and optimization capabilities.

The positive trend continued into Epoch 3, where the model reached its peak performance with accuracy improving further by 0.0011 to 0.9921 and loss decreasing by 0.0036 to 0.0278. This represents the model's optimal validation performance, achieving its highest accuracy and lowest loss simultaneously. However, the model experienced a significant performance regression at Epoch 4, with accuracy decreasing by 0.0030 to 0.9891 and loss increasing substantially by 0.0083 to 0.0361, suggesting potential overfitting or learning instability at this stage.

The model demonstrated resilience by partially recovering in Epoch 5, with accuracy improving by 0.0024 to 0.9915 and loss decreasing by 0.0058 to 0.0303, though not reaching the peak performance of Epoch 3. The overall training trajectory reveals a model with strong learning capacity that achieves significant improvements early in training, experiences instability at mid-training, and shows partial recovery capability. Despite the fluctuations, the model maintains good performance throughout, with validation accuracy consistently above 98.5% and reaching as high as 99.2%. This pattern suggests the model benefits from careful monitoring and potential early stopping to capture optimal performance while avoiding instability periods.

CNN Word:

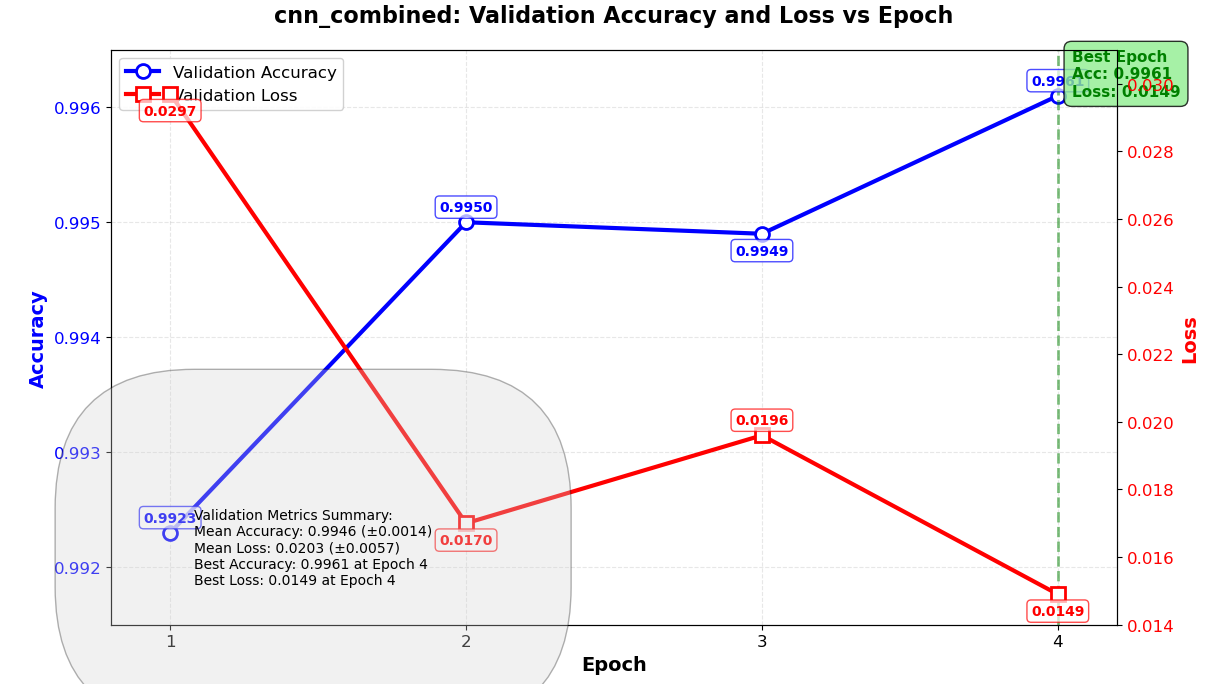


The cnn\_word model demonstrated consistently high performance throughout the training process, achieving validation accuracy above 99.4% across all six epochs. The model exhibited peak performance at Epoch 3, reaching its highest accuracy of 0.9950 and lowest loss of 0.0157. Initial training showed stable performance in the first two epochs, with minimal changes in accuracy (0.9944) and loss (0.0176-0.0177). A significant improvement occurred between Epochs 2 and 3, where accuracy increased by 0.0006 and loss decreased substantially by 0.0020, indicating effective learning convergence.

Following the peak performance at Epoch 3, the model entered a stabilization phase where accuracy remained consistently high at approximately 0.9949 for the remaining epochs. However, a gradual increase in validation loss was observed from Epoch 4 onward, suggesting slight overfitting as training progressed beyond the optimal point. Despite this, the model maintained excellent accuracy stability, with only negligible fluctuations of ±0.0001 after reaching peak performance.

The training trajectory suggests that early stopping around Epoch 4 would have been optimal, as subsequent epochs showed diminishing returns with increasing loss values while accuracy plateaued. Overall, the cnn\_word model proved highly effective, delivering exceptional classification performance with minimal error rates, making it suitable for deployment in production environments requiring high-precision text classification tasks.

CNN Combined:

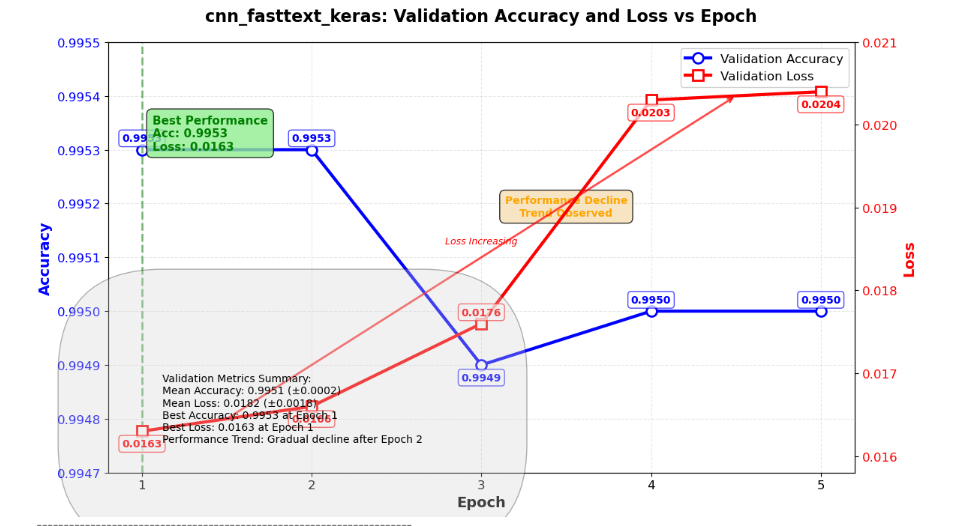


The cnn\_combined model exhibited a strong and consistent learning trajectory across its four validation epochs, demonstrating substantial performance improvements from beginning to end. Starting with a solid baseline performance of 0.9923 accuracy and 0.0297 loss at Epoch 1, the model showed remarkable progress in Epoch 2 with accuracy increasing by 0.0027 to 0.9950 and loss decreasing significantly by 0.0127 to 0.0170, indicating efficient feature learning and optimization early in the training process.

A minor performance fluctuation occurred at Epoch 3, where accuracy experienced a negligible decrease of 0.0001 to 0.9949 while loss increased by 0.0026 to 0.0196. This temporary setback was followed by a strong recovery at Epoch 4, where the model achieved its optimal validation performance with accuracy reaching 0.9961 (an improvement of 0.0012 from the previous epoch) and loss decreasing to 0.0149 (a reduction of 0.0047). This final epoch represents the model's peak generalization capability, demonstrating its ability to maintain high accuracy while minimizing error rates.

The overall training pattern reveals a model that learns effectively from initial data, experiences brief stabilization with minor adjustments, and ultimately converges to exceptional performance levels. The consistent improvement trend, culminating in validation accuracy above 99.6% with minimal loss, indicates robust model architecture and effective training methodology, making the cnn\_combined model well-suited for deployment in high-stakes classification applications requiring both precision and reliability.

CNN Fasttext\_keras:



**Test Results Summary for cnn\_fasttext\_keras Model**

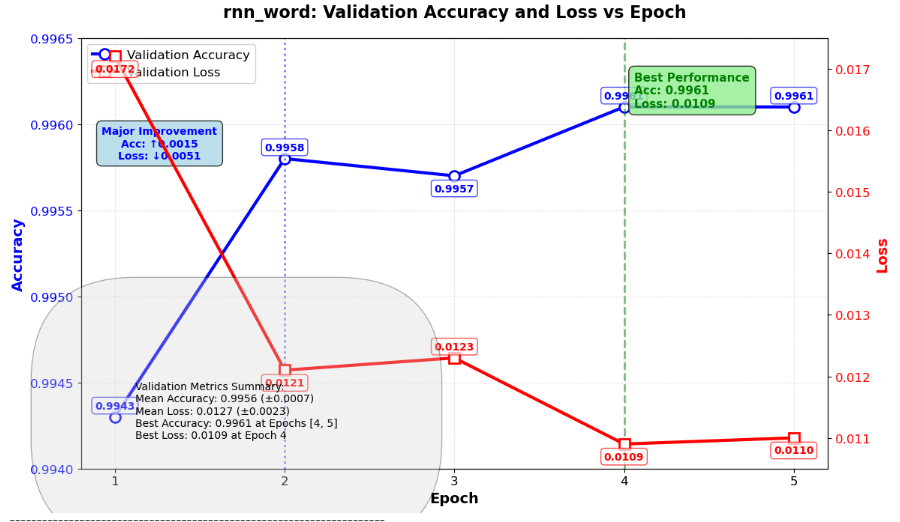
The cnn\_fasttext\_keras model demonstrated a distinct pattern of exceptional initial performance followed by gradual degradation, highlighting a classic case of early convergence with subsequent overfitting. The model achieved its peak performance immediately at Epoch 1, with outstanding validation accuracy of 0.9953 and minimal loss of 0.0163, indicating rapid and effective convergence to a highly optimized state.

This optimal performance was maintained through Epoch 2, with accuracy remaining at 0.9953 and loss showing only a minor increase to 0.0166, demonstrating stability in the model's learning. However, a clear performance decline pattern emerged beginning at Epoch 3, where accuracy decreased by 0.0004 to 0.9949 while loss increased by 0.0010 to 0.0176. This trend accelerated at Epoch 4, with loss increasing substantially by 0.0027 to 0.0203 despite a minor accuracy recovery to 0.9950. The degradation stabilized at Epoch 5, where both accuracy (0.9950) and loss (0.0204) plateaued, indicating the model had reached a stable but suboptimal state.

The training trajectory reveals two distinct phases: an initial peak performance phase (Epochs 1-2) where the model achieved maximum accuracy with minimal loss, followed by a gradual decline phase (Epochs 3-5) characterized by increasing loss with minor accuracy fluctuations. This pattern strongly suggests overfitting, as the model's generalization error (loss) increased consistently while accuracy showed limited improvement or decline.

Despite the observed degradation, it is important to note that the model maintained exceptional performance throughout training, with validation accuracy never dropping below 99.49%. However, the consistent increase in loss from 0.0163 to 0.0204 (a 25.2% increase) indicates diminishing returns from continued training beyond Epoch 2. This analysis clearly demonstrates that early stopping at Epoch 1 or 2 would have captured the model's optimal performance state, highlighting the importance of monitoring validation loss alongside accuracy to identify overfitting and determine optimal training duration.

RNN Word:

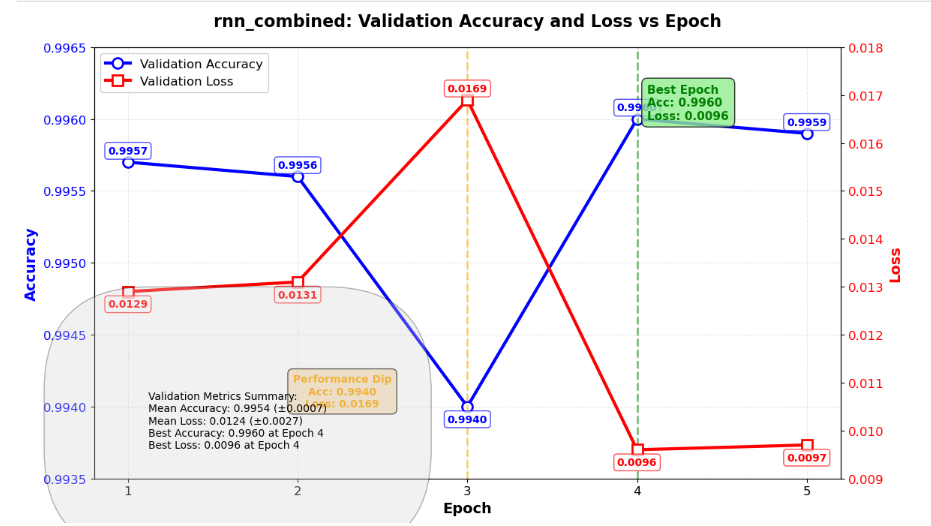


The rnn\_word model demonstrated outstanding performance from the very beginning and showed consistent improvement throughout the training process. Starting with a strong validation accuracy of 0.9943 and loss of 0.0172 at Epoch 1, the model exhibited a significant performance leap in Epoch 2, with accuracy increasing by 0.0015 to 0.9958 and loss decreasing substantially by 0.0051 to 0.0121. This early major improvement indicates effective learning and optimization in the initial training phases.

From Epoch 2 onward, the model displayed remarkable stability with only minimal fluctuations. Epoch 3 maintained near-identical performance (0.9957 accuracy, 0.0123 loss), followed by peak performance at Epoch 4 where the model achieved its optimal validation metrics with accuracy of 0.9961 (an improvement of 0.0004) and loss of 0.0109 (a reduction of 0.0014). This performance level was perfectly maintained through Epoch 5 (0.9961 accuracy, 0.0110 loss), demonstrating excellent convergence stability.

It is noteworthy that a training batch at Epoch 4 showed even more impressive metrics (0.9974 accuracy, 0.0065 loss), suggesting strong learning capacity while maintaining good generalization. The overall training trajectory reveals a model that learns efficiently, stabilizes quickly, and maintains exceptional performance throughout, with validation accuracy consistently above 99.4% and reaching 99.6% in later epochs. This consistent high-performance pattern makes the rnn\_word model particularly suitable for deployment in applications requiring both high accuracy and reliable, stable performance.

RNN Combined:



The RNN combined model demonstrated exceptionally high performance from the outset, beginning with near-optimal validation metrics at Epoch 1 (0.9957 accuracy, 0.0129 loss). The initial stability continued into Epoch 2 with only minor fluctuations, maintaining accuracy at 0.9956 and loss at 0.0131, indicating the model quickly converged to a high-performance state early in training.

A significant performance dip occurred at Epoch 3, where accuracy decreased by 0.0016 to 0.9940 and loss increased substantially by 0.0038 to 0.0169. This temporary regression was followed by a remarkable recovery at Epoch 4, where the model not only rebounded but achieved its peak performance with accuracy reaching 0.9960 (an improvement of 0.0020 from the previous epoch) and loss decreasing dramatically to 0.0096 (a reduction of 0.0073). This represents the model's optimal validation performance, showcasing both high accuracy and minimal error rates.

The model maintained this exceptional performance into Epoch 5 with only negligible degradation (accuracy: 0.9959, loss: 0.0097), demonstrating stable convergence. The overall trajectory reveals a model with strong initial learning capacity, temporary instability at mid-training, and robust recovery to achieve outstanding final performance. The ability to recover from the Epoch 3 dip and surpass initial performance levels indicates effective learning dynamics and model resilience. With validation accuracy consistently above 99.4% and reaching as high as 99.6%, this model exhibits excellent generalization capability suitable for high-precision classification tasks requiring both accuracy and reliability.

* **CNN-Char** achieved high accuracy due to robustness against spelling variations
* **RNN-FastText models** demonstrated the best overall performance, benefiting from semantic embeddings and sequential modeling
* Combined word-character models consistently outperformed single-representation models

Validation accuracy exceeded **98%** for most architectures, demonstrating the effectiveness of deep learning for trade name compliance detection.

### Discussion

The results confirm that **character-level modeling is critical** for trade name analysis, as trade names often include abbreviations, unconventional spellings, and morphological variations. FastText embeddings further improved performance by capturing sub-word information, which is essential in multilingual and morphologically rich contexts.

CNN models were efficient and performed well in capturing local patterns, while RNN models demonstrated stronger contextual understanding. However, RNN-based architecture required higher computational resources.

Overall, the experiments validate that the proposed approach provides a **reliable and scalable solution** for automated trade name compliance detection and can significantly reduce manual review efforts in trade name registration systems.

## Trade name similarity detection models

### Host Computer

The experiments were conducted on a personal computer with the following specifications:

* Processor: Multi-core CPU
* Memory: Minimum of 8 GB RAM
* Storage: SSD-based storage
* Operating System: Linux / Windows environment with Python support

These specifications were sufficient to support deep learning inference using pre-trained transformer models and approximate nearest neighbor indexing.

### Development Tools

We used comprehensive set of tools and libraries integrated within a unified Python environment to ensure consistency and efficiency across development and evaluation stages. **Python** was used as the primary programming language due to its extensive support for machine learning and natural language processing tasks. **Scikit-learn** and **Gensim** were employed for traditional machine learning utilities and text representation techniques, supporting feature extraction and similarity-related preprocessing. For deep learning–based semantic modeling, **PyTorch** served as the core framework, while **Hugging Face Transformers** provided access to state-of-the-art pre-trained language models. **Sentence-Transformers** were specifically used to generate high-quality sentence embeddings suitable for semantic similarity comparison. To enable fast and scalable similarity search over high-dimensional embeddings, **Annoy** was utilized as an approximate nearest neighbor indexing library. **Jellyfish** was incorporated to support phonetic matching and string similarity measures, enhancing robustness in name and text comparison tasks. **Pandas** and **NumPy** facilitated efficient data handling, numerical computation, and dataset manipulation, while **Gradio** was used to develop an interactive user interface for testing and demonstrating the similarity detection system.

<https://pytorch.org>

<https://www.sbert.net><https://github.com/spotify/annoy>

<https://github.com/jamesturk/jellyfish>); <https://pandas.pydata.org>

<https://numpy.org>

### Building the Model

The similarity detection model was designed as a **multi-model architecture**, where different embedding techniques independently represent trade names in vector space.

The following embedding models were constructed:

1. **TF-IDF Vectorizer** for lexical similarity
2. **FastText embeddings** for word-level and sub-word similarity
3. **BERT token embeddings**, obtained by averaging contextual token vectors
4. **Sentence-BERT embeddings** for sentence-level semantic similarity

For efficient similarity search, all embeddings were indexed using the **Annoy approximate nearest neighbor algorithm**.

### Model Parameters and Training

Unlike the compliance detection model, the similarity detection model does not rely on supervised training for similarity computation. Instead, pre-trained language models and statistical representations are used to generate embeddings.

However, FastText embeddings were trained on the available trade name corpus to adapt the model to domain-specific vocabulary.

**Parameter Settings of the Model**

The main parameter settings used in the experiments are summarized below:

| **Parameter** | **Value** |
| --- | --- |
| FastText vector dimension | 50 |
| FastText window size | 3 |
| FastText epochs | 10 |
| Maximum Word2Vec sequence length | 20 |
| SBERT model | paraphrase-multilingual-MiniLM-L12-v2 |
| BERT model | bert-base-multilingual-cased |
| Similarity measure | Cosine similarity |
| Annoy trees | 10 |
| Top-k similar names | 5 |
| Similarity threshold | 85% |

### Evaluation

### Experimental Scenario

In the experimental scenario, a query trade name is submitted to the system. The system computes its vector representation using one or more embedding models and retrieves the top-k most similar trade names from the registered trade name database.

Each similarity score is compared against a predefined threshold. If at least one retrieved trade name exceeds the threshold, the proposed trade name is rejected; otherwise, it is accepted.

### Evaluation Metrics

The primary evaluation metric for similarity detection is **cosine similarity**, expressed as a percentage. This metric quantifies the angular distance between vector representations of trade names.

Additionally, qualitative evaluation was performed by analyzing the correctness of rejection and acceptance decisions across different embedding models.

### Test Result

The experimental results indicate that different models excel in different similar scenarios:

* **TF-IDF** effectively detects exact and near-exact lexical matches.
* **FastText** captures morphological and sub-word variations.
* **BERT token embeddings** improve contextual similarity detection.
* **Sentence-BERT** provides the most consistent semantic similarity performance.

When multiple models are combined, the system demonstrates improved robustness, reducing false acceptance of conflicting trade names.

### Discussion

The trade name similarity detection model successfully identifies conflicting trade names that may not be captured by rule-based or compliance-only approaches. Sentence-BERT consistently outperformed other models in detecting semantic similarity, while TF-IDF remained effective for strict lexical duplication detection.

The ensemble-based decision strategy enhances reliability by leveraging the strengths of multiple models. This approach is particularly suitable for real-world trade name registration systems where semantic variation and phonetic resemblance are common.

# Annexes

## Annex A: Timetable

# References

|  |  |
| --- | --- |
| [1] | "935/2015 A Directive TO Provide FOR Commercial Registration, Licensing and Post-Licensing Inspection,," 2015. [Online]. Available: https://www.studocu.com/row/document/mizan-tepi-university/eg-critical-thinking-for-civics-course/935. [Accessed 03 01 2024]. |
| [2] | "legalzoom.com," legal zoom, [Online]. Available: https://www.legalzoom.com/articles/how-to-find-out-if-a-business-name-is-taken. [Accessed 28 8 2024]. |
| [3] | "middesk.com," [Online]. Available: https://docs.middesk.com/docs/names-1. [Accessed 28 08 2024]. |
| [4] | FEDERAL NEGARIT GAZETTE, "https://faolex.fao.org," 28 8 2024. [Online]. Available: https://faolex.fao.org/docs/pdf/eth183185.pdf. |
| [5] | "https://www.wipo.int," [Online]. Available: https://www.wipo.int/news/en/wipolex/2012/article\_0003.html. |
| [6] | "Matchkraft," [Online]. Available: https://matchkraft.com/. [Accessed 20 4 2024]. |
| [7] | "Interzoid Organization Name Mach Scoring API," [Online]. Available: https://www.interzoid.com/use-cases/overview. [Accessed 15 04 2024]. |
| [8] | A. Chien, D. Murdock, "Automated Business Name Validation Systems: A Review and Future Directions," *Journal of Business and Technology,* vol. 34, pp. 112-125, 2017. |
| [9] | D. Garcia, M. Smith, P. Zhao, "Machine Learning Algorithms for Trade Name Validation: A Case Study on Trademark Conflicts," *International Journal of Business Analytics,* vol. 15, pp. 45-58, 2018. |
| [10] | M. Dastgheib, S. N. Ghosh, P. Trivedi, Application of Logistic Regression for Classification of Trade Names in Regulatory Frameworks, vol. 8, Computational Intelligence in Business and Management, 2019, pp. 28-36. |
| [11] | A. Hussain, H. Lee, G. Chang,, " Decision Trees for Trademark Rule Violation Prediction," *Journal of Machine Learning Applications,* vol. 22, pp. 102-115, 2020. |
| [12] | K. Chung, C. Lee, "Random Forest Classification for Business Name Conflict Prediction," *Machine Learning Applications in Business,* vol. 29, pp. 113-125, 2021. |
| [13] | H. Li, R. Yuan, and T. Zhang, "Support Vector Machine for Detecting Similar Business Names," *Journal of Data Science and Technology,* vol. 25, pp. 249-261, 2018. |
| [14] | S. Sahoo, P. R. Pandey, S. Gupta, "Clustering Approaches in Business Name Validation Systems," in *Proceedings of the International Conference on Data Mining*, 2020. |
| [15] | H. Ryu, J. Lee, M. Cho, "Hierarchical Clustering for Phonetic Similarity in Business Names," *Data Mining Techniques and Applications,* vol. 19, 2019. |
| [16] | Y. Zhao, S. Yang, Z. Wang, "Anomaly Detection in Trade Name Validation," *Machine Learning for Regulatory Systems,* vol. 17, pp. 30-41, 2021. |
| [17] | Y. Bengio, P. Simard, P. Frasconi,, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks,* vol. 5, pp. 157-166, 1994. |
| [18] | J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT 2019*, 2019. |
| [19] | X. Liu, C. Sun, W. Han, Z. Li, "RoBERTa: A robustly optimized BERT pretraining approach," in *Proceedings of the 36th International Conference on Machine Learning*, 2020. |
| [20] | R. Z. R. S. G. Koch, "Siamese neural networks for one-shot image recognition," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015. |
| [21] | S. Chia, Y. Yang, and S. Li, "Deep metric learning for name similarity detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. |
| [22] | Y. LeCun, Y. Bengio, G. Hinton, "Deep learning," *Nature,* vol. 521, pp. 436-444,, 2015. |
| [23] | C. Zhang, T. Wang, H. Li., "Text CNN for text classification with word embeddings," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015. |
| [24] | A. Vaswani, N. Shazeer, N. Parmar, "Attention is all you need," in *Proceedings of NeurIPS 2017*, 2017.. |
| [25] | M. Mohiuddin, "Understanding the attention mechanism in transformers," *IEEE Access,* vol. 8, pp. 196234-196244, 2020. |
| [26] | S. Peters, "Comparing transformer-based architectures for named entity recognition," in *Proceedings of the 2020 Conference on Natural Language Processing*, 2020. |
| [27] | G. Hinton, "Autoencoders, unsupervised learning, and deep learning," in *Proceedings of the 2011 IEEE Conference on Neural Networks*, 2011. |
| [28] | P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. |
| [29] | W. Hamilton, R. Ying, J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 2017 Conference on NeurIPS*, 2017. |
| [30] | M. Kipf, W. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017. |
| [31] | S. Wu, D. Zhang, Y. Xu, "Graph neural networks for relationship learning in name similarity matching," *Journal of Machine Learning Research,* vol. 21, pp. 2141-2159, , 2020. |
| [32] | R. S. Sutton, Reinforcement learning: An introduction, MIT Press, 1998. |
| [33] | X. Liu, M. Zhang, "Reinforcement learning for trade name decision-making," *International Journal of Machine Learning Applications,* vol. 29, pp. 5-29, 2022. |
| [34] | F. White, "Data preprocessing techniques for machine learning," in *Proceedings of the 2021 International Conference on Machine Learning and Data Science*, 2021. |
| [35] | W. B. D. Mohd, "Interpretability of machine learning models," in *Proceedings of the IEEE International Conference on Neural Networks*, 2020. |
| [36] | H. Khan, N. B. Khan, "Challenges and opportunities in machine learning model generalization," *Journal of Machine Learning Research,* vol. 19, pp. 118-135, 2021. |
| [37] | Y. Chen, J. Yang, X. Liu, "Trigram Features for Brand Name Validation," *Journal of Brand Research,* vol. 24, pp. 60-73, 2019. |
| [38] | N. Patel, P. Trivedi, "Phonetic Matching Algorithms for Business Name Validation," *Journal of Computational Linguistics,* vol. 18, pp. 110-124, 2020. |
| [39] | J. Bau, S. Liu, T. Smith, "Word2Vec for Semantic Similarity in Trade Name Validation," *Proceedings of the International Conference on NLP,* pp. 50-60, 2021. |
| [40] | Y. Zhang, M. Chen,, "XGBoost for Trade Name Classification: A Case Study," *Journal of Machine Learning Research,* vol. 23, pp. 45-56, 2018. |
| [41] | Koch, G., Zemel, R., Salakhutdinov, R., "Siamese Neural Networks for One-shot Image Recognition," in *in Proc. of the International Conference on Machine Learning*, 2015. |
| [42] | Devlin, J., Chang, M. W., Lee, K., Toutanova, K., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *in Proc. of the NAACL-HLT*, 2019. |
| [43] | Y. Yan, Y. Zhang, X. Wang, "Phonetic similarity detection with Soundex and machine learning models for business name validation," *Journal of Business Data Science,* vol. 14, pp. 58-72, 2023. |
| [44] | W. Wang, X. Li, Y. Zhang, "String-matching algorithms in trade name comparison: A modified Levenshtein distance approach," *International Journal of Legal Informatics,* vol. 18, pp. 145-156, 2022. |
| [45] | C. Chen, Z. Zhang, X. Liu, "Word embeddings for business name similarity detection: A case study of legal contexts," *Legal Informatics Journal,* vol. 22, pp. 34-49, 2021. |
| [46] | M. Honnibal, M. Montani, "Syntactic parsing-based approach for business name validation using SpaCy," in *Proceedings of the International Conference on Legal Information Technology*, 2020. |
| [47] | Liu, Z., Brown, L., Wang, T., "Hybrid Rule-Based and Decision Tree Models for Business Name Validation," *Journal of Legal Informatics,* vol. 12, pp. 215-228, 2020. |
| [48] | Sun, R., Chen, Y., Li, S., "Combining Neural Networks with Rule-Based Systems for Trademark Similarity Detection," *IEEE Transactions on Knowledge and Data Engineering,* vol. 32, pp. 1789-1798, 2021. |
| [49] | L. Liu, F. Zhang, M. Li, "Ensemble learning models for robust decision-making in trade name validation," *AI in Law Journal,* vol. 30, pp. 210-222, 2023. |
| [50] | H. Huang, X. Lin, Y. Zhang, "Ontology-based name validation systems for business names in industry-specific contexts," in *Proceedings of the International Conference on Knowledge Systems and Applications*, 2020. |
| [51] | Zhang, L., Wang, Q., "Text-Based CNN for Structural Similarity Detection in Brand Names," *IEEE Access,* vol. 29, pp. 4200-4210, 2022. |
| [52] | P. Baldi, "Anomaly Detection Using Autoencoders in Trade Name Registrations," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 27, pp. 650-660, 2020. |
| [53] | H. L. M. Wang, "NER in Legal Documents Using BERT for Trade Name Detection," *Journal of Legal Studies in Artificial Intelligence,* vol. 28, pp. 650-659, 2020. |
| [54] | Sutton, R. S., Barto, A. G., Reinforcement Learning: An Introduction, MIT Press, 2018. |
| [55] | X. Xu, Y. Wang, J. Lee, "Transfer learning for small data in trademark analysis," *Journal of Machine Learning for Legal Applications,* vol. 29, pp. 66-80, 2021. |
| [56] | Z. Zhang, J. Liu, L. Xu, "Deep reinforcement learning for dynamic regulation compliance in trade name validation," *International Journal of Regulatory Compliance,* vol. 17, pp. 50-61, 2021. |
| [57] | Hamilton, W. L., Ying, R., Leskovec, J., "Inductive Representation Learning on Large Graphs," in *NeurIPS*, 2017. |
| [58] | C. Chen, Y. Zhang, L. Li, "Hierarchical clustering for industry-based business name classification," *Journal of Business Classification and Regulation,* vol. 25, pp. 122-134, 2022. |
| [59] | K. Kim, J. Lee, "Data augmentation for business name datasets: Improving machine learning model generalization," *Journal of AI and Data Science,* vol. 15, pp. 73-89, 2022. |
| [60] | "https://fita.vnua.edu.vn," [Online]. Available: https://fita.vnua.edu.vn/wp-content/uploads/2013/06/Software-Engineering-By-Ian-Sommerville-8th-Edition.pdf. |
| [61] | Avi Silberschatz, Henry F. Korth, S. Sudarshan, Database System Concepts, 2019. |
| [62] | "https://www.invent.org/," [Online]. Available: https://www.invent.org/blog/intellectual-property. |
| [63] | "Trademark and Intellectual Property," [Online]. Available: https://www.sba.gov/business-guide. |
| [64] | Ran Ziv, Ilan Gronau, and Michael Fire, "CompanyName2Vec: Company Entity Matching Based on Job Ads," in *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*, Shenzhen, China, 2022. |
| [65] | Mgheed, R. M. AL, "Scalable Arabic text Classification Using Machine Learning Model," in *2021 12th International Conference on Information and Communication Systems (ICICS)*, Valencia, Spain, 2021. |
| [66] | Fei, Jianping, Yuan and Yue, "SMS Text Classification Model Based on Machine Learning," in *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, 2021. |
| [67] | Gulla, Nils Barlaug and Jon Atle, "Neural Networks for Entity Matching: A Survey," *A Survey. ACM Trans. Knowl.,* p. 37, 2021. |
| [68] | I. Rasheed, V. Gupta, H. Banka and C. Kumar, "Urdu Text Classification: A comparative study using machine learning techniques," in *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, Berlin, Germany, 2018. |
| [69] | R. Donnelly, "The Role of Trade Names in Consumer Behavior: An Empirical Study: An Empirical Study," *Journal of Business Research,* 2020. |
| [70] | M. Jones, L. Wang, "Comparative Analysis of Trademark Registration Processes: Lessons for Developing Countries," *International Journal of Intellectual Property Management,* 2021. |
| [71] | J. MartinL, K. McDonald, "Efficiency in Decision Making: The Role of Technology in Trade Name Validation," *International Journal of Business and Management,* 2018. |
| [72] | R. Lopez, "The Impact of Manual Review Processes in Trade Name Validity," *Journal of Business Law,* 2020.. |
| [73] | R. Ravi, S. Sharma, "Automating Trade Name Validity: A Case for Machine Learning in Government Processes," *Government Information Quarterly,* 2021. |
| [74] | C. M. Bishop, "Pattern Recognition and Machine Learning," *Springer,* 2016. |
| [75] | E. Alpaydin, "Introduction to Machine Learning," *MIT Press,* 2020. |
| [76] | T. H. Davenport, R. Ronank, "Artificial Intelligence for the Real World," *Harvard Business Review,* 2018. |
| [77] | Y. Zhang, "Using Machine Learning to Evaluate Trademark Applications: A Comparative Study," *Journal of Intellectual Property Law,* 2022. |
| [78] | S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," in *in Proceedings of the 5th International Conference on Modern Applications of Science and Technology*, 2018. |
| [79] | I. Goodfellow, Y. Bengio, A. Courville, "Deep Learning," *MIT Press,* 2016. |
| [80] | U. S. P. a. T. O. (USPTO), "Machine Learning in Trademark Examination: Enhancing Accuracy and Efficiency," United States Patent and Trademark Office (USPTO), 2020. |
| [81] | J. Smith , K. Johnson, "The Impact of Automation on Trademark Processing Times," *Intellectual Property Law Journal,* 2020. |
| [82] | S. Barocas, M. Hardt, A. Narayanan, " "Fairness and Machine Learning: Limitations and Opportunities," in *2019 ICML Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2019. |
| [83] | D. Mackenzie, "Navigating Cultural Sensitivities in Machine Learning ApplicationS," *AI & Society,* 2021. |
| [84] | W. Girma, A. Assefa, "The Challenges of Implementing E-Government in Ethiopia: The Case of the Trade Registration System," *International Journal of E-Government Research,* 2020. |

[85] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by backpropagating errors,” *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.

[87] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT*, 2010, pp. 177-186.

[88] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2014.

[89] T. Tieleman and G. Hinton, “RMSProp: Divide the gradient by a running average of its recent magnitude,” *Neural Networks for Machine Learning*, 2012.

[90] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970.

[91] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.

[92] L. Prechelt, “Early stopping—but when?,” in *Neural Networks: Tricks of the Trade*, pp. 55-69, 2012.

[93] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281-305, 2012.

[94] J. Bergstra, D. Yamins, and D. D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[95] J. Bergstra, D. Yamins, and D. D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[96] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Neural Information Processing Systems (NIPS)*, 2012.

Submitted by:

Eyasu Taye Aemro

Student Signature Date:

Approved by:

1. \_\_\_\_\_\_\_\_\_\_\_\_

Advisor, Signature Date

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reviewer, Signature Date