Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# SCAN-NF: a Machine Learning System for Invoice Product Trasaction Classification Through Short-Text Processing.

Diego Santos Kieckbusch

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Li Weigang

Brasília
2022

## Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB.  A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

# Esta página não deve ser inclusa  na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# SCAN-NF: a Machine Learning System for Invoice Product Trasaction Classification Through Short-Text Processing.

Diego Santos Kieckbusch

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Li Weigang (Orientador)
CIC/UnB

Profa. Dra. Anne Magály de Paula Canuto
DIMAp/UFRN

Prof. Dr. Geraldo Pereira Rocha Filho
CIC/UNB

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Programa de Pós-graduação em Informática

Brasília, 10 de Novembro de 2022

# Dedicatória

Eu dedico esta obra a minha familia, meus pais Andrea e Rafael, por terem me mostrado o valor do estudo, do trabalho duro e da busca por conhecimento e excelencia como um fim em si mesmo. Dedico também a minha futura esposa, Katarine, por toda compreemsão, suporte e por sempre acreditar que tudo daria certo.

# Agradecimentos

Agradeço a todos aqueles que possibilitaram a realização desta pesquisa. Primeiramente a meu orientador Professor Doutor Li Weigang pelas oportunidades fornecidas, ao Professor Geraldo pela atenção no acompanhamento da escrita dos trabalhos. Agradeço também aos colegas do Translab pelas recorrentes discussões. Agradeço aos colegas Sergio e Vinicius, cuja colaboração nos compartilhamentos dos dados e exposição do problema original possibilitaram esta pesquisa.

Agradeço, novamente, minha família e amigos por todo o apoio, compreensão e paciência. Por criarem o ambiente que me permitiu concluir este trabalho.

# Resumo

Nota Fiscal Eletrônica (NF-e) é um documento que reporta as transações de bens e serviços de forma eletrônica, tanto na transferência quanto no armazenamento. A utilização de notas fiscais eletrônicas é uma tendencia emergente e apresenta uma valiosa fonte de informação para diversas áreas. No entanto, o processamento dessas notas é uma tarefa desafiadora. A informação reportada está geralmente incompleta ou apresenta erros. Antes que qualquer processamento significativo possa ser feito, é necessária identificar o produto representado em cada documento. A literatura disponível indica que são necessárias arquiteturas especializadas para lidar com este tipo de informação. Este trabalho propõe SCAN-NF, uma arquitetura para a classificação das transações de produtos contidas em notas fiscais eletronicas. A arquitetura modela o problema de processamento de notas ficais como um problema de processamento de textos curtos com o objetivo de identificar o produto de cada transação. A solução tem o intuito de auxiliar as tarefas de auditoria manual feita por auditores fiscais sobre grandes massas de dados não rotulados ou mal rotulados presente no contexto de notas fiscais. Para validar a arquitetura proposta, este trabalho apresenta tanto um framework contextual para o processamento de notas fiscais quanto um caso de estudo utilizando dados reais de notas fiscais. Modelos tradicionais baseados em frequência de termos foram comparados a modelos de classificação de sentenças baseado em redes convulsionais artificiais. Experimentos demonstram que embora o texto presente em notas fiscais seja breve e apresente erros e falhas de escrita, modelos simples baseados em frequência de termos apresentam bons resultados para a etiquetagem de código de produtos, atingindo acurácia de até 98% entre as classes de produtos estudadas. Mostramos ainda, que é possível a utilização de transferencia de conhecimento entre os dados de notas fiscais destinadas ao consumidor e notas fiscais de transações entre empresas.

**Palavras-chave:** Aprendizado Profundo, Redes Convolucionais, Classificação de textos curtos

# Abstract

An electronic invoice (E-invoice) is a document that records the transactions of goods and services electronically, both in storage and exchanges. E-invoice is an emerging practice and presents a valuable source of information for many areas. Processing these invoices is often a challenging task. Information reported is often incomplete or presents mistakes. Before any meaningful processing of these invoices, it is necessary to identify the product represented in each document. The available literature indicates that specialized architectures are necessary to deal with this type of information. This work proposes SCAN-NF, an architecture for invoice product transaction classification. The architecture models the invoice classification problem as a short-text classification problem, in which the goal is to identify the type of product in each transaction based on its short-text description. This solution is intended to aid tax auditors in the analysis of large unlabeled or poorly labeled invoice data. To validate the proposed architecture, this work provides both a contextual framework for invoice processing and a study case utilizing real-world invoice data. We compare traditional term frequency models to sentence classification models based on convolutional neural networks. Experiments demonstrate that even though invoice text descriptions are brief and present many mistakes and typos, simple term frequency models can achieve high baseline results on product code assignment, reaching accuracy scores up to 98% in studied product classes. We have also shown that it is possible to utilize transfer learning between retail invoice data and business to business invoice data.

**Keywords:** Deep Learning, Convolutional Neural Networks, Short-text Classification, Transfer Learning

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Invoices document the transactions of goods and services between two parties. Invoicing is a core component in daily commercial and financial operations. They are a rich source of information for financial analysis, fraud detection [2], value chain analysis, product tracking, and hazard alarms [3]. Extracting useful information from invoice documents can lead to valuable applications. However, processing invoices is a difficult task due the scale and nature of the data. Text in invoices is often brief and presents poor grammar. This associated with the variety of products makes rule-based processing unfeasible. This work tackles the problem on how to automatically identify the product in each transaction contained in electronic invoices, based on a short text description present in each transaction. We present SCAN-NF, a machine learning system to aid tax auditors in processing information contained in electronic invoices.

The initial question presented was: how can we discover fraudulent behavior in brazilian invoices? By consulting with tax auditors, it was noted that one of the main ways in which issuers evade taxes is to purposely miss-classify the type of taxes to be applied to each product transaction. The type of tax is closely tied to the type of product. If the product type is known, the correct taxes can be identified trough additional business rules. In retail invoices there is no auditing over the fulfillment and correctness of the product code, which identifies the type of the product. The problem then becomes how to correctly identify the NCM code for each product transaction. We framed the problem as a short text classification problem and looked for Machine learning techniques that could use the large amount of data stored in the state treasury office to create a intelligent system to aid tax auditors.

Product code assigns each product to a specific class in a taxonomy based on the type of product. In this work we utilize the Common Mercosur Nomenclature (NCM) to classify

each product. The NCM code is used as reference in assigned taxation and other policies as wells as being a common reference on product classification. While NCM is particular to countries in Mercosur, functionally taxonomy's may be available in other countries. Correctly assigning the NCM code is the first step towards more complex analysis. In this work we provide a contextual framework to guide developers ans researchers through the possible applications.

In Brazil, e-invoicing process started in 2008, first with NF-e, and latter with the NFC-e, which is a nationwide transaction reporting integrated system for both business to business (B2B) and retail operations. Similar measures have also been taken in Italy and China [4][5]. As of 2022, every transaction of goods and services in Brazil should emit electronic invoice in the form of a xml file to the corresponding treasure office. The physical document that accompany many products is just a auxiliary document (DANFE), the invoice itself is the xml file reported to the treasure office server. This makes Brazil an excellent study case, as the schema for invoice data have already been standardized.

Recent emerging techniques in the field of Machine Learning (ML) and Natural Language Processing (NLP) have allowed valuable applications. This methods allows the models to learn how to do a particular task through training on data. This would solve the problem of rule-based classification, as instead of the manually creating rules for a large amount of classes taking hours of effort form specialists, models can be built on available reported data. In this work we take a look at different forms of representation as well ad different classification models. We train Support vector models on term based representation and different Artificial Neural Network models on word-vector representation on NCM classification task. Models are trained and tested on real world data provided by the state treasure office in a study case on Brazilian Electronic Invoice (NF-e), and Brazilian Consumer Electronic Invoice (NFC-e). This documents report both B2B transactions and retail transactions respectively.

While there is a wide array of possible tasks that can utilize invoices as input, in this work, we assume the point of view of the treasure office. The treasure office has access to a large stream of electronic invoices. One of its many tasks is to check for fraud in collected taxes. Once a suspicious issuer is identified, an auditor is assigned to conduct a deep look into that issuer finances. This process is costly and there is a balance between the amount of effort and the amount of tax funds that could be recuperated. It would be of great value if a automated process could point out suspicious or fraudulent invoices and flag them for human review. This is expected to increase the productivity of tax auditors.

Figure 1.1 presents an overview of invoice processing in three phases. At the bottom, Label 1, we have both retail and larger companies that issue invoices as part of their day-to-day activities. In Figure 1, these invoices are represented by the NF-e and NFC-

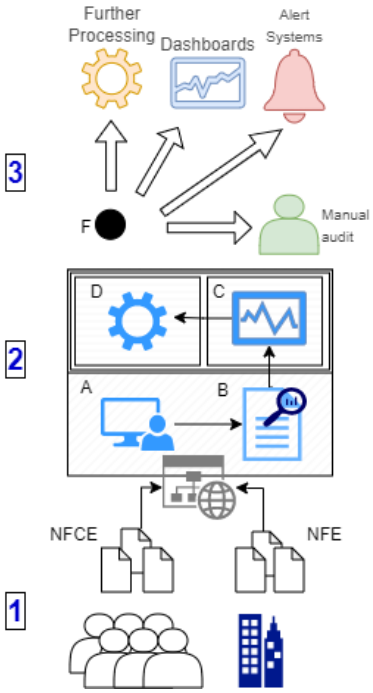



Figure 1.1: Invoice Processing in a Nutshell

e documents, the Brazilian document for retail, and B2B invoices. These invoices are reported to a centralized system through web applications. Once reported, these invoices are processed to aid in a particular task. This process is depicted in Label 2, as an analyst selects relevant data to the core problem (A), data is then cleaned (B), explored (C), and used as the input to train a task-specific model, Label D. The trained model and analyzed data set (F) is then used as input for other applications and to aid manual auditing of other invoices, Label 3.

Modelling fraudulent behavior is a complex subject that may take input from many indicators. While the are some automated routines for checking validity of reported fields in the NF-e documents, there is still room for improvement, specially in the processing of NFC-e. Retail invoices are not checked at creation and possess inconsistencies and missing fields. Before any more complex tasks can be done, such as fraud detection [2], value chain and health hazards triggers [3], we have to correctly identify what products are represented in each invoice. The problem of invoice classification becomes correctly identifying the product being referred to in that transaction.

According to the literature, electronic invoices are a particularly difficult problem for short-text processing. Even compared to other short text, such as Twitter posts and news headlines, the invoice product description is very brief, containing only a handful of words, often not forming a complete sentence. Another difference, is that invoice classification is not a natural language problem. Short texts are often produced by instant human com-

munication in the form of micro-blogs, tweets, and news. There is an intrinsic attempt at creating a communication channel with others. Product descriptions are defined individually with no regard to how that message is perceived by the other end. This exacerbates the problem of domain-specific vocabulary, abbreviations, and typos, as authors use their individual logic.

In recent years, works on product-level invoice classification have concentrated in China. Their solutions range from using hashing techniques to dealing with an unknown number of features [4][5], semantic expansion trough external knowledge bases [5], classification of paragraph embedding by k-nearest-neighbors [6] to different artificial neural network architectures [7][8]. Semantic expansion is prevalent not only on invoice classification but also on short-text classification [9],[10]. These works are not suited for the Brazilian case either due to language differences or reliance on knowledge bases that are often only available in English and Chinese [11]. In the literature, there are gaps in the models suitable for classifying languages other than Chinese.

As the main contribution of this work, we present the architecture for SCAN-NF, a system for labelling product according to the Mercosur Common Nomenclature (NCM) based on the short text description present in product transactions. The system is intended to aid tax auditors on invoice processing tasks, in which errors in reported transactions may indicate fraud. In order to validate SCAN-NF's approach, we present both a contextual framework for invoice processing and a study case on product level classification of invoices based on Brazilian invoice data. We present experiments using character-level, word-level CNN and support vector machines. Character level representation could be useful to tackle typos and abbreviations, such tokens would not be correctly represented when using pre-trained word embedding. Support Vector machines trained over term frequency-inverse document frequency representation act as an example of a term-count model. Our case study focuses on invoices in Brazil, since case study data was obtained through cooperation with the state treasury office.

## 1.2 Hypothesis

Text has been avoided in processing NF-e and NFC-e data. Product descriptions are supposedly too brief and to poor to be used as input. We hypothesize that it is possible to utilize short text processing and few-word classification techniques to classify each product in regards to the Mercosur Common Nomenclature (NCM). This classification method would allow a system to aid tax auditors by identifying the correct type of product for each transaction.

## 1.3 Objective

In order to test our hypothesis, we will provide an architecture for a machine learning system to aid tax auditors. This model will be utilize a machine learning model to classify invoice product transactions. The model will take the product description field of the NF-e and NFC-e documents as input and output the corresponding NCM code for that product. Model will be tested on transfer learning based on how models trained with one type of document could be used to support the other type of document.

### 1.3.1 Specific Objectives

- Provide an architecture for the classification of short texts with few words.
- Evaluate the proposed architecture on NFC-e product code classification task.
- Compare the architecture to state of the art NLP methods.
- Compare models on transfer learning task.
- Present a real use case for the trained model.

## 1.4 Additional Contributions of This Work

In addition to this document, this research has produced the following contributions to the academic community:

- SCAN-NF: A CNN-based System for the Classification of Electronic Invoices through Short-text Product Description [12].
- Towards Intelligent Processing of Electronic Invoices: the General Framework and Case Study of Short Text Deep Learning in Brazil. Publication pending on Springer Series: Lecture Notes in Business Information Processing.

## 1.5 Dissertation Structure

This document is organized as follows: chapter 2 presents to the reader the fundamental concepts of text mining, short text processing, Neural Networks and Transformers. In chapter 3, related works on short text classification and invoice classification are discussed. Chapter 4 presents the overall system architecture, describes both NF-e and NFC-e documents, the classification task, the architecture of the proposed model, and

the planned experiments. Chapter 5 presents our implementation of the models. In chapter 6 experiment results are presented. In Chapter 7, we present conclusions and future work.

# Chapter 2

# Fundamentals

The goal of the chapter is to present fundamental concepts for the understanding of this work to the reader. Section 2.1 presents the concepts associated with text mining. Section 2.2 describes the characteristics of short text and how early work attempted to address the problem. Section 2.3 presents basic concepts in deep learning, with segments for feed-forward artificial neural networks (ANN), convolutional neural networks (CNN), and recurrent neural networks (RNN). We also present an explanation for the additive attention mechanism. Section 2.4 presents popular word embedding algorithms such as Word2vec, Glove, FastText, and how these embeddings can be evaluated and combined. Finally, section 2.5 presents a brief overview of the transformer model.

## 2.1 Text Mining

Text mining refers to the extraction of useful information from text with the aid of various statistical algorithms. It may also be referred to as text analytics and machine learning for text [13]. Due to the large expansion of the internet and digital communications, the field has increased in popularity. Modern sources of text data include digital libraries, electronic news, web applications, and social media. Two possible feature representations are popularly used in text mining applications: text as a bag of words and text as a set of sequences. Bag-of-Words representation is based on the frequency of terms, disregarding grammar, and word order. When treating text as a set of sequences, language-related properties are preserved. Text mining has traditionally focused on the first type of representation, but advances in artificial intelligence have allowed for easier modeling of language semantics, reasoning, and understanding through artificial neural networks. Traditionally, these features were handcrafted and were task dependant, this made them time-consuming and costly. Deep Learning allowed for automatic feature engineering of multi-task language features through neural language models [14].

In the bag of words (BOW) representation, the corpus is represented by a document-term matrix. Each row represents a document and each column represents the frequency of a term in that document. Since only a small part of the vocabulary is present in each document, the final representation is a sparse, non-negative, high dimensional matrix. Many of the techniques applied in text mining aim at reducing the dimensions of the document-term matrix through feature engineering or filtering. Topic model methods such as Latent Dirichlet allocation (LDA), Latent Semantic Analysis (LSA) aim at clustering features into topics based on the co-occurrence of terms.

Artificial neural networks (ANN) utilize a different approach for feature engineering. In artificial neural networks, text is represented as a sequence of vectors. These vectors are usually initiated through pre-trained word embeddings. Document representation is learned in conjunction with the classifier through supervised learning. Neural based text processing will be further discussed in section 2.3.

The typical pipeline for text mining includes the following steps: data collection, data preprocessing, feature engineering/filtering, classification/clustering, and evaluation. Text co-occurs with a lot of extraneous data such as tags, anchor text, and application-related features. Furthermore, text needs to be converted from its unstructured form to a structured and multidimensional representation.

### 2.1.1 Pre-processing

Text preprocessing includes text extraction, stop-word removal, stemming, case folding, and frequency-based normalization. Text is often created by human participants in unstructured environments. It is often embedded in web documents, with application-specific tags, misspellings, and ambiguous words. The goal in preprocessing is to extract tokens from the original raw data.

Tokens are sequences of characters that are treated as indivisible units in text. For each word in the document, a token is computed. After prepossessing, tokens are converted into terms with specific frequencies that represent the collection. While a document can have repetitions of the same token, terms consolidate these sets of tokens into a single occurrence with an appropriate frequency.

Words with high frequency that are common across documents provide little information about the document's content. These words are called stop-words and can be removed from the collection of terms. All articles, prepositions, pronouns, and conjunctions are stop words. Any token that is too frequent can also be considered a stop-word if its frequency surpasses a manually defined threshold. Language-specific dictionaries can be used to identify stop words. An alternative to stop-word removal is through down-weighing frequent words in the normalization step.

In most cases, words can be converted to their lowercase form. Nonetheless, capitalization may occur for different reasons, such as differentiation of proper nouns and common nouns and verbs. The token *"Rosa"* could be related to a person's name, while *'rosa'* would refer to the color or the flower. This would result in two different terms. A collection of simple rule base heuristics can be utilized in casing, such as words at the start of a sentence can always be converted to lower case, or words in titles and headers can be converted to lower case. The case for all other words is retained.

In order to reduce variance in the data, small variations of the same token could be represented by the same term. Stemming is the process of consolidating related words with the same morphological root. For example, singular and plural forms of the same noun can be consolidated into a single term. The same is true for verbs in multiple tenses. Most common techniques for stemming include semi-automatic lookup tables, suffix stripping, and lemmatization.

Lookup tables are created in advance through various heuristics, linking different tokens to a singular term. Suffix stripping stores a small list of rules in order to find the root of a word by removing suffixes. Lemmatization goes beyond simple stripping rules and uses morphological domain-specific knowledge to identify *lemmas*. Lemmatization also requires part of speech tagging in order to produce the correct result. Python Natural Language Toolkit [1] provides stemmers for several languages, including Portuguese.

### Vector Space Representation and Normalization

The outputted terms from preprocessing form the *lexicon* that is used as the base set of dimensions. However, not all terms provide the same amount of information. Normalization allows us to identify the most significant terms instead of using the raw term count. Tf-IDF(*term frequency-inverse document frequency*) is a popular form of normalization. It normalizes the raw term frequency based on the inverse document frequency:

$$\begin{aligned} tfidf &= tf \cdot idf \\ idf &= log(\frac{n}{n_i}) \end{aligned} \tag{2.1}$$

where $tf$ is the term frequency, $n$ is the total number of documents and $n_i$ is the number of documents in which the term appears. At the limit, where $n = n_i$, the value of $idf$ is 0. There are variations to tf-idf based on functions of $tf$, such as $tf^2$ and $log(1+tf)$. Although its common practice to utilize tf-idf normalization, some application may perform better with raw frequencies or binary values [13].

[1] available at https://www.nltk.org/

### 2.1.2 Dimensionality Reduction

A high number of dimensions leads to more data being needed to fit models. Furthermore, many classification algorithms struggle with sparse data. This motivates efforts for the creation of a lower dimensionality representation for text. In the document-term matrix, term columns often correlate to one another and can be leveraged to generate a low dimensional representation of data. These dimensionality reduction techniques are based on *low-rank* factorization of the document-term matrix. Low rank factorization takes a $n \times d$ document-term matrix $D$, with $n$ documents and $d$ terms, that can be expressed in terms of $k \ll min\{n, d\}$ $d-$dimensional basis vectors. The value of $k$ defines the number of semantic concepts in the data. A $d \times k$ matrix $V[v_{ij}]$ can be constructed, that links individual $d$ terms to the basis vectors. Furthermore, documents can also be expressed in term of the basis vectors by a $n \times k$ matrix $U = [u_{ij}]$. Therefore the document-term matrix can be represented by a factorized form:

$$D \approx UV^T \tag{2.2}$$

The general idea of the factorization is that the remaining $(d-k)$-dimensional does not have significant representation in the corpus at hand and its captured by the approximate equality. The discovery of $U$ and $V$ aims to minimize residual error $(D-UV^T)$. Algorithms differ in the objective function and the constraints applied to the $UV$ matrices.

Latent Semantic Analysis (LSA) applies the constraint that the $UV$ matrices should be orthogonal, which facilitates computation through eigenvalue decomposition. Non-negative matrix factorization and Probabilistic Latent Semantic Analysis constraint to non-negative matrices, which improves the interpretability of the factorization. Latent Dirilecht Allocation assumes that each document is a mixture of topics, and each topic, in turn, is a mixture of other topics.

Another form of dimensionality reduction is through feature selection. Instead of creating a new feature space, filtering selects a subset of available features through ranking based on a score, such as raw term frequency or TF-IDF [15].

### 2.1.3 Classification

Classification is the process of assigning a document or item to a set of finite classes. Supervised classification models utilize labeled data to learn a function that maps input features to target classes. Each example is a tuple consisting of an input vector and the desired output value. Models are evaluated on their ability to correctly classify unseen data.

### K-Nearest Neighbours

K-Nearest Neighbours is a non-parametric supervised learning model, in which new data is classified by similarity to known data. Each new entry is classified based on the most common class among $k$ nearest observations. In figure 2.1, examples are shown for different $k$ values. For $k = 3$ the new white observation would be assigned to the blue class, if we utilize $k = 5$ it would be assigned the red class. The choice parameter $k$ is done through hyper-parameter optimization and is data dependant.

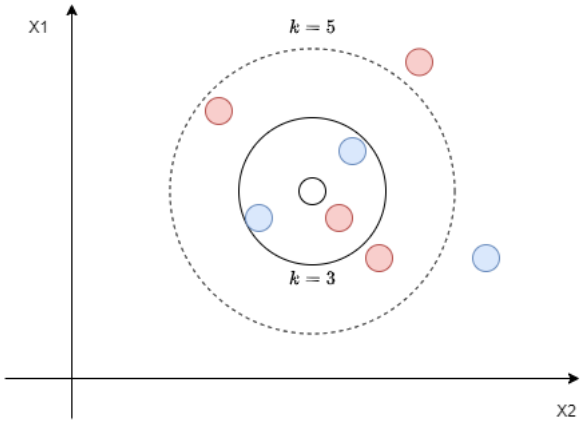



Figure 2.1: Example of K-NN classifier for k=3 and k=5

### Support Vector Machine

Support Vector Machine (SVM) is a supervised learning model that works by maximizing the margin that separates samples through a high dimensional hyperspace. Figure 2.2 presents a graphical representation of a linear SVM. The hyperplane that separates the feature space is given by:

$$g(\overrightarrow{x}) = \overrightarrow{w}^T \overrightarrow{x} + \omega_0 \tag{2.3}$$

where $\overrightarrow{w}^T$ represents the weight vector, $\overrightarrow{x}$ is the coordinate vector and $\omega$ is a constant. The goal of learning is to minimize $||\overrightarrow{w}||$ in order to maximize the margin. Fitting the support vector classifier involves minimizing the following expression:

$$minimize\left\{ \sum max[0, 1 - y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} - \omega)] + \lambda ||\overrightarrow{w}||^2 \right\} \tag{2.4}$$

where $\lambda$ is a non-negative tuning parameter that dictates the tolerance of the model to miss-classified observations. A high $\lambda$ leads to a higher tolerance and a soft margin, while a low *lambda* leads to a low tolerance a hard margin. SVM is able to handle nonlinear

classification through the use of kernel functions. Kernel function substitute the dot product between coordinates by a similarity function. The kernel allows SVM to model nonlinear functions by projecting the maximum-margin hyper plane in a transformed feature space.

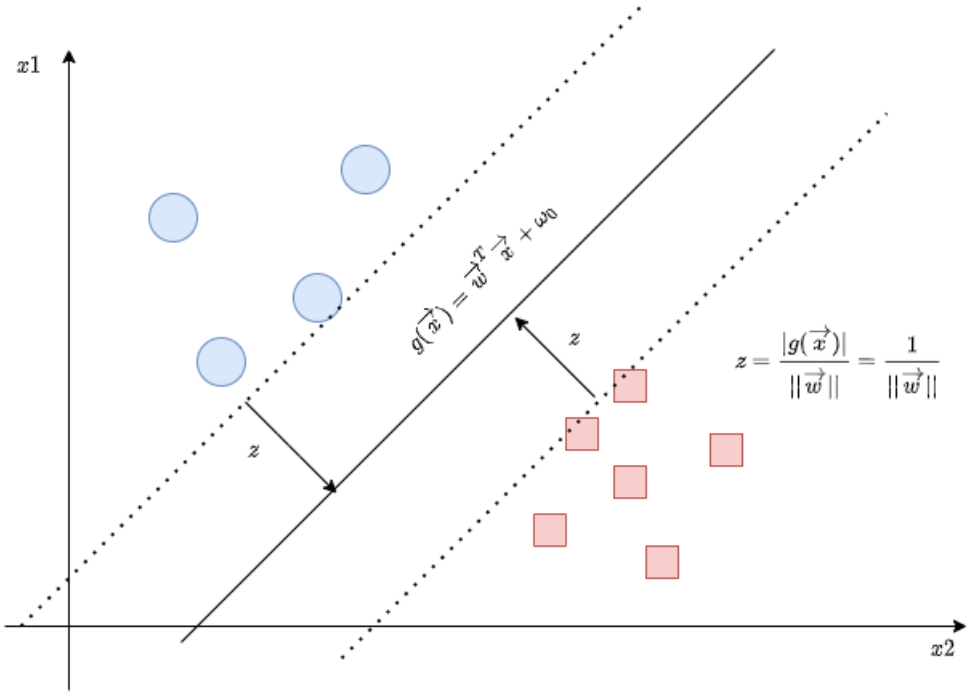



Figure 2.2: Linear Support Vector Machine

## 2.2 Short Text Processing

Short text is a type of text characterized primarily by its brevity. Traditional text mining approaches presented in section 2.1 struggles with short text processing due to low term count. Short text is deeply linked to Web 2.0, characterized by user-created content. Due to the nature of its environment, short text is also characterized by poor grammar and domain-specific vocabulary. Short text characterization can be done by [16]:

- Individual author contributions are very brief and data is sparse. It does not provide enough word co-occurrence or shared context for good similarity measures. This makes it more difficult to extract valid language features.
- Grammar used by authors is generally informal and unstructured, relative to a particular domain. There are many misspellings, non-standard terms, and noise.
- Text is semi-structured by NLP definitions since it contains some metadata [17].

- Immediacy: short-text is often sent and received in real-time and in large quantity.
- Imbalanced Distribution: Background applications deal with large amounts of short-text data. However, tasks often involve detecting a small number of objects. Therefore, useful instances are limited and the distribution of short text classes is imbalanced.
- Large scale data and labeling bottlenecks: It is expensive to manually label all data. How to better combine labeled and unlabeled instances is one key problem of text classification.

.

Short texts exist in a variety of forms: SMS messages, e-commerce reviews, instant message apps, online chat, bulletin board systems, Twitter [18]. Short text processing allows for an array of applications, such as classifying news, search queries, identifying and removing erotic content messages, classifying/clustering tweets, blog messages and scientific abstracts and sentiment analysis [18]. Manual monitoring of online content is tedious and expensive. Short text processing allows for better situational awareness for industry, business, community, and military use [19].

Traditional text mining pipeline struggles with short text processing. Due to the low word count, the document-term matrix is particularly sparse, with documents being represented only by the presence of few terms with low frequency. The abundance of abbreviations, misspellings, and dialects makes it harder to breakdown the matrix through dimensionality reduction. For the rest of this section, we will present non-neural approaches to short text processing.

Under the nomenclature of microtext, Ellen [17] surveyed AI and NLP techniques applied to military chat rooms, SMS, voice transcriptions, and micro-blogging. Early techniques did not focus on the text but tried to leverage metadata in order to classify short-text. The metadata utilized was:

- Source Attribution (Author, Screen Name, Originating Phone Number or Email Address)
- Time stamp (Almost always with minute-level accuracy)
- Audience (Public, Room or Chat channel, source attribution)
- URL References (reply/threading mechanism, longer reference)
- Geo-location information (location tags, GPS)
- Other application-specific data (hashtags, mood, weather, user created and automatically generated, such as whats-app status. )

On notable research, the author cites Twinner [20], a Twitter clustering based on physical location and TweetMotif [21] that grouped tweets based on statistically unlikely phrases that co-occur. Rosa [22] performed experiments on military chat posts with SVM's K-Nearest Neighbours, Rocchio, and Naive Bayes classifiers, using mutual information and information gain as feature selection methods. However, these methodologies performed poorly on binary and four-way classification tasks. So far, short text work could be characterized by leveraging outside bodies of knowledge and non-traditional language features to create task-specific solutions, with little effort to create generalized models for short-text.

Rafeeque [18] surveyed short text analysis. The general framework for short text analysis included two steps before classification/clustering: expansion of the sparse features with additional information from a linked long text or document, and a measure of short text similarity. The document is fetched from an external source such as a database repository, blog, file system, or the internet.

Web-based short text similarity used web documents returned by a search engine to compute the similarity between two short texts or words. Web-kernel similarity[23] and web relevance similarity [24] are computed using normalized term vectors extracted from the collection of returned documents. These similarity measures were primarily used for query suggestions, but they could also be used to create relationships between terms and entities when there is no available taxonomy.

Sriram [25] used domain-specific features of Twitter to classify tweets into five generic classes such as News, Events, Opinions, Deals, and Private Messages. A total of 8 features were used: the author name, and seven tags representing the presence of slangs, time-event phrases, opinionated words, emphasis, currency and percentage signs, usernames at the beginning or middle of the text. The engineered features outperformed BOW in the classification of 5407 random tweets of 684 authors using the Naive Bayes Classifier.

Alsmadi and Gan [15] conducted a review on short text classification. The author made a distinction between feature selection and extraction. The most common feature was a term vector with different weighting schemes, such as binary weights, term frequency, log of term frequency, Tf-IDF, and probability term frequency. Classifiers were separated between example-based, probabilistic, decision tree, and linear classifiers. Of the reviewed articles, 89% utilized feature selection., while only 11% used feature engineering. Concerning classifiers, 44% used linear classifiers, 21% used probabilistic classifiers, 14% used decision Trees, 13% used ensemble classifiers and 8% used example based classifiers.

Qiang et al. [26] conducted a survey on topic modeling techniques for short text. Traditional short text topic models such as LDA, LSA, and PLSA performed poorly on short-text due to the low co-occurrence of words in short-text. The author was able to

identify three categories of topic models: Dirichlet multinomial mixture based models, global word co-occurrence based methods, and self-aggregation methods.

## 2.3 Deep Learning

Roughly speaking, machine learning is the process of representing data in a computer-oriented manner to interactively discover patterns through parameter optimization; be it maximizing in-group similarity, while minimizing cross-group similarity in clustering; or minimizing error function in classification. These techniques are limited in their ability to process data in its raw form, requiring data representation techniques that able to extract relevant information from available data for the task at hand.

The key aspect of deep learning is that these layers of features are not designed manually. They are learned from data using a general purpose-learning procedure. Deep Learning constructs a complex representation expressed in terms of other simpler representations. While the overall model maps inputs to outputs, the function is formed by composing many simpler mathematical functions of each layer. We can think of each application of a different mathematical function as providing a new representation of the input [27].

Deep Learning has been particularly useful in tasks that demanded good representations due to the high dimensionality of data, such as image and text processing [28] [29]. Research in these fields has been traditionally oriented towards extracting features that could capture meaningful information from raw data, such as filters for images and POS tagging in text processing. Deep Learning has allowed advances in these fields and produced many applications such as object detection and speech recognition [28].

### 2.3.1 Feedforward Neural Network

Feedforward neural networks, or multi-layer perceptrons, are the quintessential deep learning models. The goal of the feedforward network is to approximate some function $f$ that maps input $x$ to a category $y$. A feedforward neural network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters $\theta$ that result in the best function approximation [27].

Goodfellow provides explanations on feedforward neural networks as a function approximator. Each of the three terms is explained individually. Feedforward comes from the fact that information flows from the input layer $x$, moves through the internal representations used to define $f$, and generates the output $y$. There are no connections in which information is fed back into itself. This contrasts with other models such as Recurrent Neural Networks, which include feedback connections.

The network portion is based on the compositionality of the final learned function. The model can be represented as an acyclic graph that describes how the functions are composed together. A 3 layer network could be described as $f(x) = f^(3)(F^(2)(f^(1)(x)))$, in which $f^(1)$ is the first layer, $f^(2)$ is the second and so forth. The number of layers in this chain is called the depth. This is where the deep part of the terminology comes from and serves to contrast with shallow linear models that do not possess the same representation power.

The neural part comes from the fact that these networks are loosely related to neuroscience. Early developments can be traced to the creation of mathematical models for pattern recognition of neurons in the perceptron algorithm and similar models [30]. The orientation of ANN research has changed through the years from an accurate representation of the brain to function approximation machines guided by mathematical and engineering techniques [27]. Nonetheless, there are still architectural decisions that are inspired by how neurons work.

Each layer in a neural network is not a single vector to vector function, but a collection of different units that take all the outputs from the previous layer and compute a single scalar each. The output of said layer will in turn be used by the following layer. The resulting scalar of each unit is bound by an activation function, similar to how neurons work. Figure 2.3 presents a network with two hidden layers and the output of each layer. Each node takes the weighted sum of the outputs of the last layer, represented by $z$, and produces an output $y$ based on the activation function $f(z)$ that represents the neuron activation.

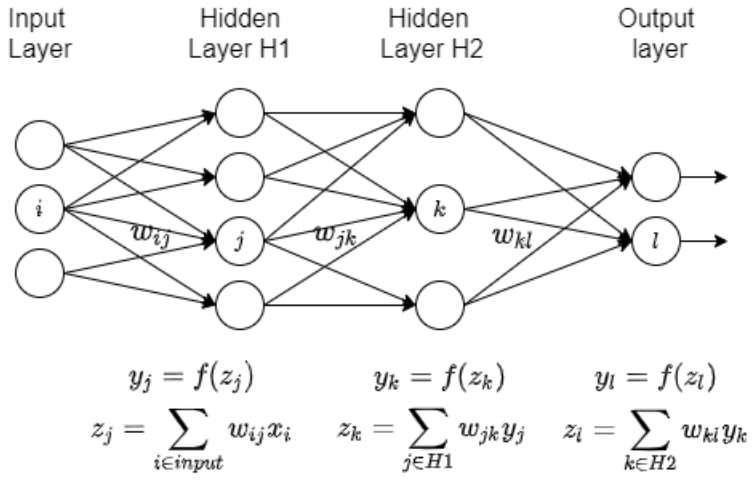



Figure 2.3: Feedforward Neural Network. Adapted from [28]

The choice of activation function impacts the training of the neural network as the cost of calculating the derivative impacts back-propagation. The slope of the activation

function is also linked to the vanishing gradient problem due to how the error signal deteriorates the further it travels through the network. Figure 2.4 shows the flow of the error signal backwards through the network. The partial derivative of the error functions indicates the contribution of the term to the total error. In the example provided by Figure 2.4, the error function is given by $\frac{1}{2}(y_l - \dot{y})^2$, with $y_l$ being the network output, $\dot{y}$ the true value of the target variable. This leads to the partial derivative of the error in the output layer presented in the example. For the hidden layers, the derivative of the error is a weighted sum of the error propagated by latter nodes. The partial derivatives can be calculated through the chain rule.

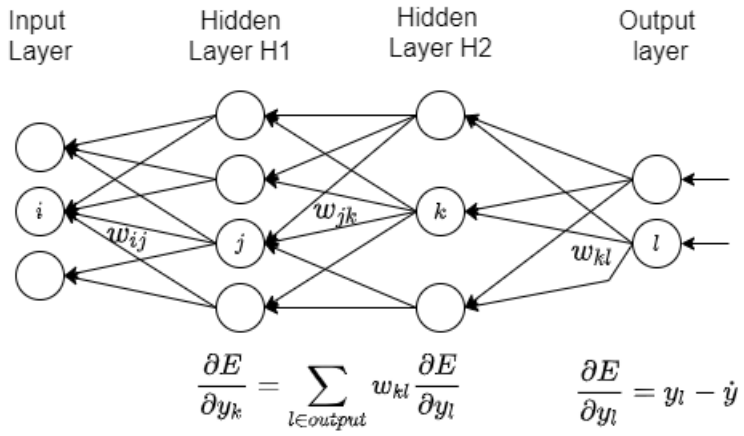



Figure 2.4: Back-Propagation of Error Signal. Adapted from [28]

Neural networks are usually trained through an iterative, gradient-based optimizer that drives a cost function towards a very low value. The non-linearity of neural networks cause loss functions to become non-convex. This means that there is no guarantee that gradient descent will converge on a global minimum. Furthermore, there are several hyper-parameters in neural network training that impact the final result, such as the learning rate, the number of epochs of training, batch size, drop-out rate, and momentum.

Learning rate is the most important hyper-parameter. Learning rate dictates the rate at which weights are updated. If the learning rate is too large, the model can converge too quickly to a sub-optimal solution, while a low learning rate may get stuck. Optimizers, such as Adam [31] and AdaGrad [32] can be employed to utilize per adaptive learning rates per parameter.

### 2.3.2 Convolutional Neural Network

Convolutional neural networks (CNN) are a specialized kind of neural network for processing data through convolutions operations. Convolutions is an operation on two functions

of real-valued arguments, Goodfellow [27] provides an example as to how convolutions can be understood in the context of neural networks. Imagine a sensor $x(t)$ that gives the position of an object at time $t$. This sensor is noisy. In order to prove a better estimate of the position of the object, we can use the convolution of the signal $x(t)$ and a weighting function $w(a)$ of the age of measurement $a$. In computation time is discrete and the convolution can be given by:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.5}$$

If the convolution is done over more than one axis, over an image $I$ for example, we can use a two-dimensional kernel K:

$$S(i,j) = (I * K)(i,j) \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.6}$$

Convolution is commutative so:

$$S(i,j) = (K * I)(i,j) \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.7}$$

In order to obtain the commutative property the kernel is flipped. Many machine Learning Algorithms do not flip the kernel, and instead utilize cross-correlation:

$$S(i,j) = (K * I)(i,j) \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.8}$$

Convolutions leverage three important ideas that can help improve machine learning systems: sparse interactions, parameter sharing, and equivariant representations. In feedforward neural networks, operations between layers are done through matrix multiplication. For every possible combination of input and output unit, there is a parameter. In convolutional neural networks, this parameter interaction is sparse, since the kernel is generally much smaller than the input, and the same kernel is applied several times over the input.

This means that fewer parameters must be trained, which both reduces the memory requirements of the model and improves statistical efficiency. If there are $m$ inputs and $n$ outputs, matrix multiplication requires $mxn$ parameters, then algorithms used in practice have $O(m \times n)$ runtime. By limiting the number of connections, the sparse connections require only $kxn$ parameters and $O(k \times n)$ runtime, with $k$ being the kernel size. In practice, $k$ is often orders of magnitude lower than $m$, resulting in a large improvement.

Sparse interactions are also related to parameter sharing. In traditional neural networks, every interaction is associated with a weight that is used only once. By reapplying

the same kernel multiple times over the input, weights associated with that kernel are shared over multiple computations. This doesn't affect the number of runtime operations but reduces the number of parameters that must be stored.

In turn, parameter sharing is related to the property of equivariance to translation. A function $f(x)$ is equivariant to another function $g$ if $f(g(x)) = g(f(x))$. Convolutions are equivariant for translation. This means if the input is shifted, the transformation will be shifted by the same amount. This is useful since the function of a small region of the matrix can be applied to multiple locations. In image classification, this means that a particular kernel that is able to detect vertical edges can be used all over the matrix to identify those same edges. In subsequent layers, this results in a filter that is capable of finding more complex patterns, such as a face, independently to where it appears in the image. It is important to notice that while convolutions are equivariant to translation, they are not equivariant to other transformations, such as rotation.

The typical convolutional layer includes the convolution stage, detector stage, and pooling stage. The convolution stage is defined by the affine transformation done by applying the kernel to the input. The detector stage is where an activation function is applied, similar to that of feedforward neural networks. The final stage is pooling. Pooling replaces the output of the network with a summary of the statistics at that location. Pooling helps to make the representation invariant to small translations of the input. This means that if we translate the input by a small amount, the values pooled do not vary significantly change. Pooling also improves the computational efficiency of the network because the next layers have fewer inputs to process.

When dealing with textual information, text is first converted to a sequence of vectors by applying pre-trained word embedding. This results in a representation of size $n \times k$, $n$ being the length of the sequence, and $k$ the numbers of dimensions of the word embedding. Figure 2.5 presents the CNN architecture propose by Kim [33] for sentence classification. Unlike images, text is a one-dimensional representation. Convolutions in text processing are 1-dimensional and slide through the sequence. Filters of different sizes are utilized in order to learn multi-word features. Convolutions are followed by a max-over time pooling that selects the maximum value of each feature over all positions in the sequence. In practice, this means that the pooling layer summarizes whether or not that feature is present in the sequence. The resulting feature vector is connected to a fully connected neural network that will behave as the classifier.

### 2.3.3 RNN Based Models

While convolutional neural networks attempt to capture local patterns in data, recurrent neural networks (RNN) attempt to capture sequential patterns. They are based on state

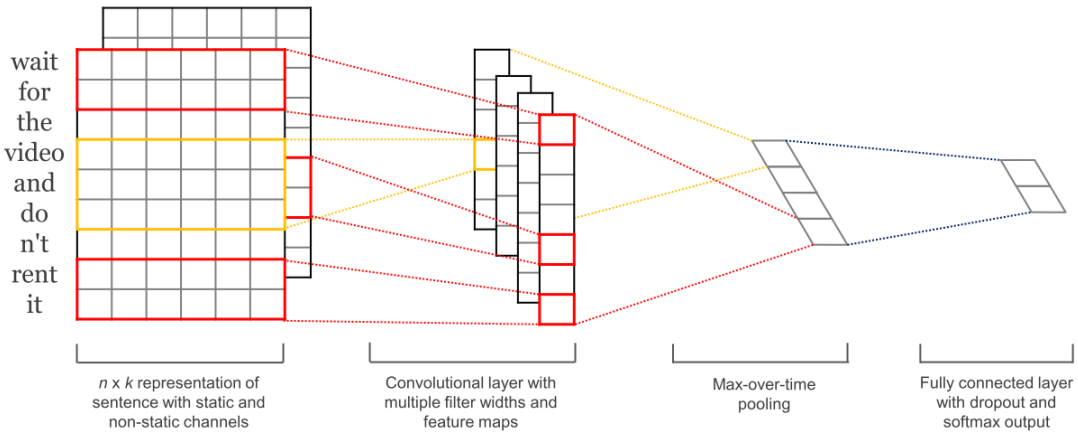



Figure 2.5: CNN For Word Classification. Extracted from [33]

machines, in which the current output of the node depends not only on the input but also on the previous state of the node.

The most basic form of RNN used is based on Elman machines [34]. Elman machines consist of input, hidden, and output layers. For the sake of comprehension, RNNs are often represented in their unfolded format in order to explicitly express relationships between variable time inputs

Since individual weight contribution is determined not only by the current input $x_t$ but also based on previous step computed hidden layer $h_{t-1}$, RNNs are particularly prone to the vanishing gradient problem. This meant that RNN could not capture long term relationships of sequences. In order to address this problem, models such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were created with the intent of preserving long-term contextual information. Figure 2.6 presents the overall architectures for these model's units.

LSTM [35][29] introduced forget gates over simple RNN architecture in order to indefinitely preserve signal. LSTM consist of three gates: input $i$, forget $f$, and output $o$ gates.

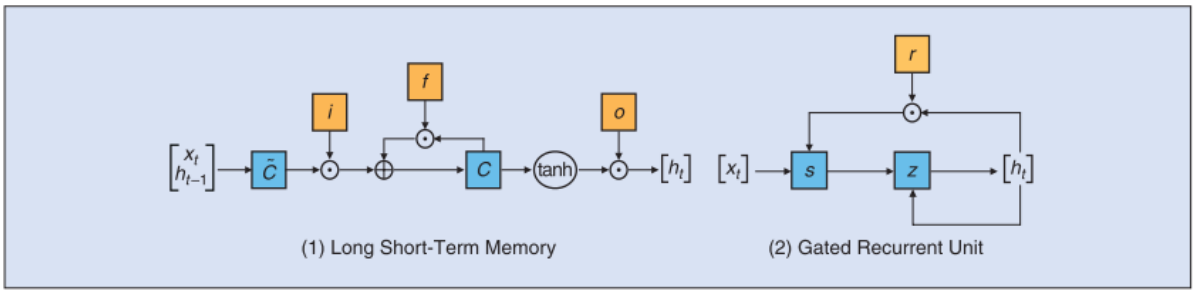



Figure 2.6: LSTM and GRU architecture, extracted from [29]

Signals in LSTM are defined by the following equations:

$$x = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \tag{2.9}$$

$$f_t = \sigma(W_f \cdot x + b_f) \tag{2.10}$$

$$i_t = \sigma(W_i \cdot x + b_i) \tag{2.11}$$

$$o_t = \sigma(W_o \cdot x + b_o) \tag{2.12}$$

$$c_t = f_t \bigodot c_{t-1} + i_t \bigodot tanh(W_c \cdot x + b_c) \tag{2.13}$$

$$h_t = o_t \bigodot tanh(c_t) \tag{2.14}$$

GRU [36] is a simpler variant of RNN based on reset and update gates. GRUs have similar performance to that of LSTMs and there has been no consensus on which is the best model [29]. GRUs are governed by the following equations, where at time $t$ the unit is presented with $x_t$ and produces output $h_t$:

$$z = \sigma(U_z \cdot x_t + W_z \cdot h_{t-1}) \tag{2.15}$$

$$r = \sigma(U_r \cdot x_t + W_r \cdot h_{t-1}) \tag{2.16}$$

$$st = tanh(U_z \cdot x_t + W_s \cdot (h_{t-1} \bigodot r)) \tag{2.17}$$

$$h_t = (1 - z) \bigodot s_t + z \bigodot h_{t-1} \tag{2.18}$$

LSTMs have been used at multiple levels in the NLP field. At word level interactions, LSTM allowed for better sentence representation by taking into account word order and relationships between more distant words. This is particularly useful for resolving negation and transition words. Traditional methods for the creation of word embeddings are based on distributional hypotheses [37] that terms with similar distribution have similar meanings. This does not hold true for negation and transition words. In tasks that require natural language understanding, such as sentiment analysis, these distinctions are important [38]. Bidirectional LSTM networks [39] utilize 2-layers of LSTM that process

data in opposite directions. Forwards states process the sequence from beginning to end, while backward states process the sequence in reverse. The output of the network is produced by the concatenation of both layers at each position. Figure 2.7 presents a representation of bidirectional RNN. Essentially, any RNN based network, such as RNN, LSTM, or GRU, can be made bidirectional.

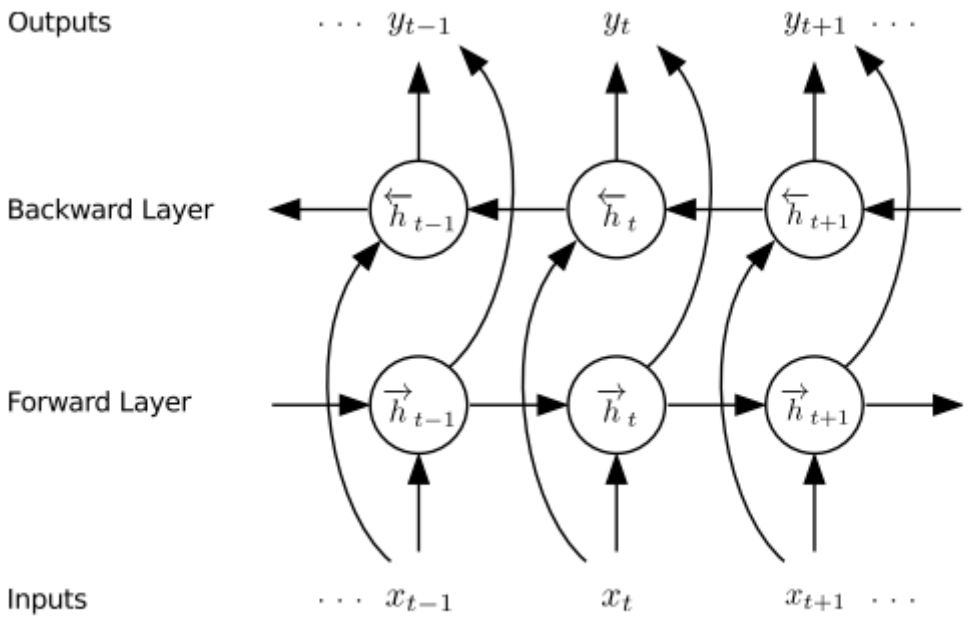



Figure 2.7: Bidirectional RNN. Extracted from [39]

### 2.3.4 Attention Mechanism

Recurrent neural networks have been widely used in sequence to sequence modeling. The overall architecture for this problem is based on the encode-decoder architecture. In encoder-decoder architecture, the network can be divided in two segments: the encoder part creates an embedding based on inputs and the decoder generates outputs based on the embedding created. Common uses for this architecture include machine translation and text generation.

While standard RNN architectures only utilize the last produced vector as sentence representation, the attention mechanism leverages hidden states produced during the processing of all time steps. The attention mechanism proposed by Bahdanau [40] creates a context vector from hidden states as a weighted sum of hidden states. Figure 2.8 presents an illustration for the attention mechanism over a bidirectional RNN. For a sentence of length $T$ :

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.19}$$

The weight $\alpha_{ij}$ of each annotation is computed by:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.20}$$

where

$$e_{ij} = f(s_{i-1}, h_j) \tag{2.21}$$

is an alignment model that scores how well the inputs around position $j$ and the output at position $i$ align. The score is based on the decoder last state $s_{i-1}$ and hidden state at position $j$. This model is implemented as a feedforward neural network that is trained in conjunction with other components. In practice, this means that the weight of each hidden state is learned by a fully connected layer that takes the hidden states as input. This generates dynamic weights across sentence positions that can track important words.

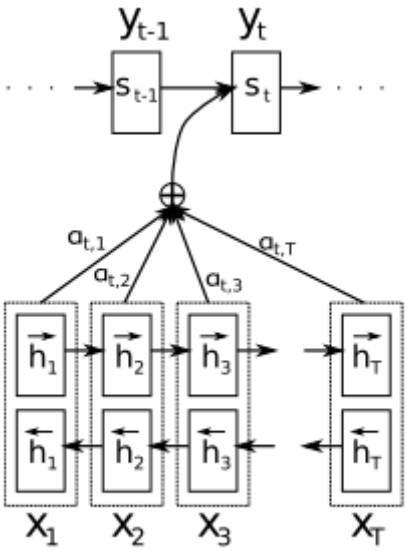



Figure 2.8: Illustration of attention mechanism on encode decode architecture. Extracted from [40]

## 2.4 Word Embeddings

Word embeddings are a group of different methods that aim to represent words in distributed vector form. Traditional word representations often scale with vocabulary size,

as the document-term matrix grows. By representing a word in a continuous-distributed representation, expansion of the vocabulary does not increase the representation size. Attempts at creating word representation through neural networks date back to the early stages of neural networks [41]. These early attempts were limited by the availability of data and computational capacity of the time. More recent models such as N-gram neural language model [42] are trained over corpora of millions of words. In this section, we will present popular word embeddings such as Word2vec, Glove, and fastText.

### 2.4.1 Word2vec

Word2vec is a C++ library for computing two different neural language models based on the work of Mikolov [43][44]. Mikolov proposed both Continuous Bag of Words model(CBOW) and the Skip-Gram model, represented in Figure 2.9. CBOW predicts a word based on it's surrounding, while skip-gram predicts the context of a center word. Models were trained on the Google news dataset, containing 6 billion words, with vocabulary restricted to the most frequent 1 million words. Training complexity is proportional to:

$$O = E \times T \times Q \tag{2.22}$$

where $E$ is the number of training epochs, $T$ is the number of words in the training set and Q is related to each model. CBOW $Q$ training complexity is given by:

$$Q = N \times D + D \times \log_2(V) \tag{2.23}$$

Where N is the context length, V is vocabulary size and D is the number of nodes in the projection layer. The projection layer differs from the traditional hidden layer used in neural networks in the sense that it does not utilize an activation function. The idea is that the reduced representation power is offset by lower computational cost. This allows the model to be trained with a larger corpus that produces better results. These shallow neural networks are referred as log-bilinear models and are represented in Figure 2.10

For the skip-gram model Q training complexity is given by:

$$Q = C \times (D + D \times log_2(V)) \tag{2.24}$$

where $C$ is the maximum distance of prediction. In training, a $R$ number in range $< 1; C >$ is selected, and the model makes predictions of the next and the former $R$ words.

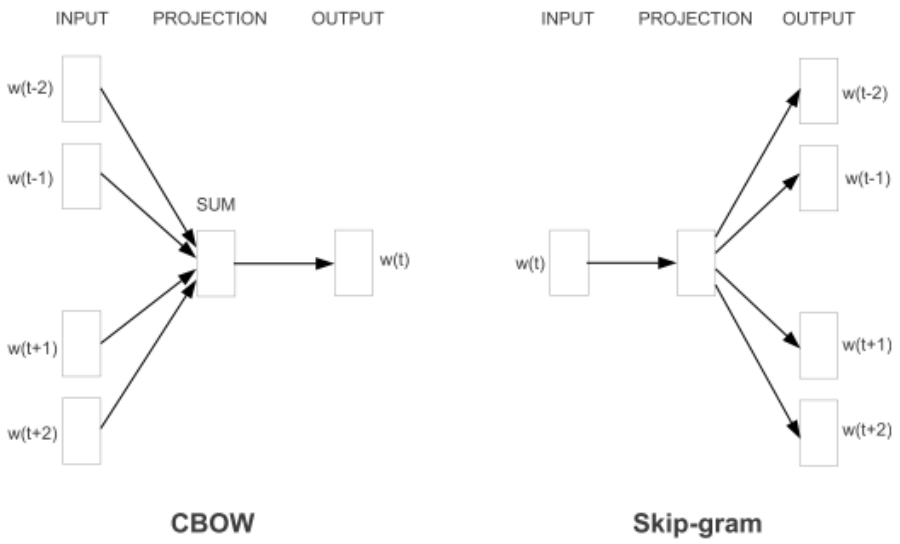



Figure 2.9: CBOW and Skip-Gram models. CBOW predicts words based on context and Skip-gram predicts context based on a given word. Extracted from [43]

.

### 2.4.2 FastText

FastText is a word embedding library based on improvements over CBOW model [46]. Improvements involve: position dependant weighting, n-gram representation, and subword information [47]. Position dependant weighting relates to the learning of weights attributed to each position of the context window. In CBOW the context vector is simply computed from the average of word vectors in the context.

FastText addresses n-grams during the pre-processing stage. during the pre-processing stage the word2phrase tool from the word2vec project is used to merge 50% of occurrences of words with high mutual information. This process is repeated up to six times. This way, words like "New" and "York" can be merged in n-grams like "New_York", "New_York_University", and so on.

The most important difference is that fastText utilizes subword representation. Words are represented as the combination of n-grams of characters. Each n-gram has its own vector representation. The final word representation is a sum of the learned representation and the n-grams vectors related to that word. This is particularly useful to treat misspells and provide representation for morphological similar words.

For the English language, word vectors were trained using a 9.2 billion token Wikipedia corpus and a 630 billion token Common Crawl corpus. These[2] and word embeddings for

[2]https://fasttext.cc/docs/en/english-vectors.html

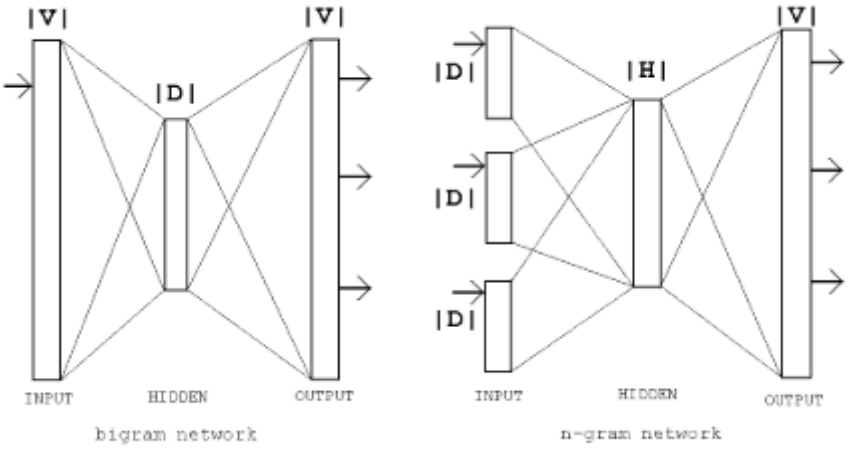



Figure 2.10: Word2vec architectures where based on shallow neural networks. Extracted from [45]

other 157 languages were made available online[3].

### 2.4.3 Glove

Since word2vec models are trained only on small context windows, Glove [48] aims to incorporate global co-occurrence in the training of word vectors. The idea is that the ratio of co-occurrences is more important than the raw number of co-occurrences. The author provides examples for the words "Ice" and "Steam". While the co-occurrence of both words to the word 'Solid' is very small, the co-occurrence between ('Ice', 'Solid') is several times that of ('Steam', 'Solid'). In turn, the co-occurrence of ("Ice", "Gas") is several times smaller than that of ('Steam', 'Gas'). Only in the ratio, does the noise from non-discriminate words cancel out. If we exchange "Gas" or "Solid" for words such as "Water" or "Fashion", the ratio will be close to 1, as both of these words are equally related or disconnected to "Ice" and "Steam".

Be $X$ the word co-occurrence matrix, and $X_{ij}$ the number of times the word $j$ occurs in the context of word $i$. $X_i = \sum_k X_{ik}$ is the number of times any word appear in the context of $i$ and $P_{ij} = P(j|i) = X_{ij}/X_i$ the probability that the word $j$ appears in the context of the word $i$.

The ratio between $P_{ik}$ e $P_{jk}$ depends on the words $i, j, k$ so the model must take the form $F(w_i, w_j, \tilde{w})$. The model is trained resolving a least square regression problem in the form of :

[3]https://fasttext.cc/docs/en/crawl-vectors.html

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{2.25}$$

Like word2vec, word vectors created by Glove also manage to capture latent semantic relationships between pairs of words through euclidean distance. While the distance between two vectors does not equate to the same underlying semantic relationship, this relationship can be contextualized by the distance of other word pairs. Such is the case for the distance between "man" and "woman" being similar to the distance of "boy" and "girl".

Glove was trained on five corpora of different sizes: 2010 Wikipedia dump of 1 billion tokens, 2014 Wikipedia dump of 1.6 billion tokens, Gigaword 5 with 4.3 billion tokens, the combination of Wikipedia and Gigaword for 6 billion tokens, and 42 billion tokens from Common Crawl. Corpora were tokenized by the Stanford Tokenizer, with a vocabulary of the 400,000 most frequent words. Training is split between populating the co-occurrence matrix and training the model. These pre-trained embeddings were made available, including additional embeddings based on Twitter [4].

### 2.4.4 Evaluation

Word embeddings can be evaluated in two ways: intrinsic and extrinsic evaluation [49]. Intrinsic evaluation utilizes human-annotated word pairs as a representation of words relationship. Evaluation is done on how well word embeddings match these pairs. Extrinsic evaluation utilizes performance metrics on downstream tasks to rank word embeddings. A downstream task is any task that utilizes the representation provided for the word embedding as input, e.g POS, chunking, or classification.

For a time, intrinsic representation was thought to be a good predictor of extrinsic performance. Intrinsic evaluation is easier and faster to perform and doesn't rely on task-specific metrics. However, research showed that good performance on intrinsic evaluation did not correlate to extrinsic performance [49].

Word embeddings trained as part of a supervised task capture semantic relationships relevant to that particular task and may differ from the relationship captured by unsupervised training. A word embedding trained on POS tagging may present similar representations for words such as 'man' and 'cat' as they are both nouns. This same embedding would be penalized by intrinsic evaluation [50].

The overall solution for word embedding evaluation has been to use extrinsic evaluation. Extrinsic evaluation is task-dependant, this makes it difficult to generalize perfor-

[4] https://nlp.stanford.edu/projects/glove/

mance evaluation of word embeddings. Nevertheless, there have been efforts for creating evaluation tool-kits in the pursuit of universal sentence representation [51].

### 2.4.5 Meta Embedding

The difference in the semantics captured by different embeddings motivated research on how to combine these embeddings. Meta embeddings are not trained on a pre-selected corpus; instead, they are created by the combination of different pre-trained embeddings. The two main advantages of meta embeddings are performance enhancement and word coverage [52].

Yin and Schütze [52] proposed 4 methods for meta-embedding computation in increasing levels of complexity. First, there is the simple concatenation of word vectors, second is the single value decomposition of the concatenated form of word vectors. The third model, named *1toN*, treats individual embeddings as projections of the meta embedding vector. The meta embedding vector is randomly initiated and the learning objective is to minimize the euclidean distance to the sum of projections. The fourth model, named *1toN+*, is aimed at vocabulary filling of out of word vocabulary.

Despite its simplicity, simple concatenation of word embeddings proved to be a good baseline for the performance of meta-embeddings [52]. Simple averaging of word embeddings also produced comparative good results and had better performance for word analogy tasks [53].

Kiela et al [54] argue that instead of generic operations over word embedding, it is simpler and more effective to allow neural networks to decide which embedding to use based on performance on the downstream task. The author trained meta embeddings not only on language inference tasks but also on multi-modal learning, based on both text and image data. Aside from performance, the main benefit of dynamic embeddings is that the individual performance of word vectors can be explained by the learned weights.

## 2.5 Transformers

Prior to transformers, RNN based architectures, such as LSTM and GRU, were the state of the art for sequence modelling. However, due to the intrinsic cost of sequential computation, there were an interest in models that could be better parallelized and take advatange of GPUs.

Figure 2.11 presents the transformer architectur, Transformers follows the encode-decode architecture, in which for a given sequence of symbols, the encode part generates a continuous representation the encapsulates that sequence. The decode part of the model utilizes this representation to generate a new sequence of symbols. Transformers

differs from other sequence transduction models by substituting the processing units in the encode-decode architecture , such as convolutional layers and lstm cells, for attention units. In order to model the sentence structure, model adds a positional embedding to the input embedding.

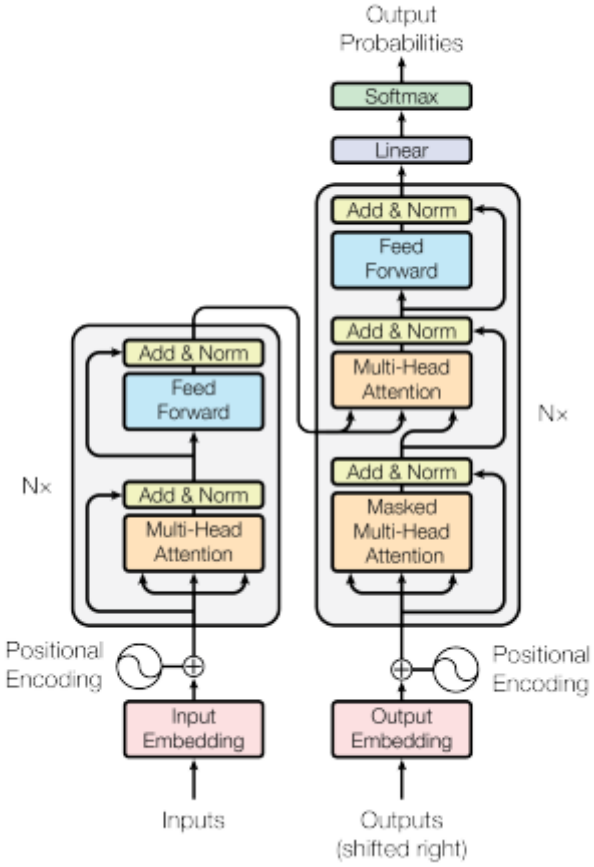



Figure 2.11: Original Transformer Architecture. Extracted from [55]

The original transformer model employed two types of attention, as shown in figure 2.12. The attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}V)$ . The self attention units are processed in parallel though the multi-head attention unit.

## 2.6 Discussion and Review

This chapter presented the relevant concepts regarding text classification, short-text processing, deep-learning, word-embedding and transformers. Traditional text classification rely on term-frequency in order to create a vector representation for text. This represen-

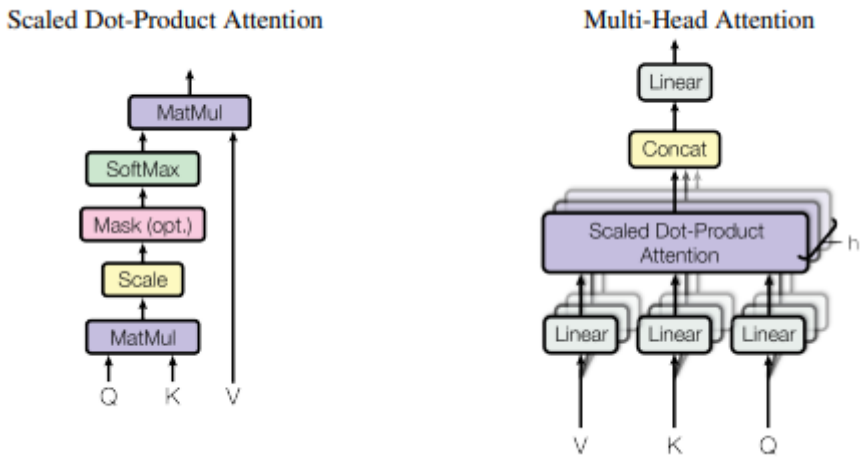



Figure 2.12: Attention Mechanism in the transformer model. Extracted from [55]

tation is high dimensional and sparse due to each document using only a small portion of the vocabulary. Short-text is a special type of text, it's main characteristic is briefness and poor quality. Due to its briefness, traditional term-frequency methods struggle with short-text related tasks. Deep learning differs from traditional methods by preserving the word order. In deep learning text is represented by a series of vectors and a representation is learned from the sentence structure or vector sequence. It is possible to initialize these vector with vectors pre-trained on a different task in order to provided semantic information. This vector may be combined to provided different representation for the same text. Transformer allowed for model pre-trained on a much larger corpus by allowing parallelization of training. Transformer are able to preserve context information for longer sentences than other sentence-based models. Due to the short-comings of traditional methods, deep-learning based methods have been investigated as possible solutions to invoice classification problem. The next chapter presents how different works addressed short-text classification and invoice classification though both traditional and deep-learning methodologies.

# Chapter 3

# Related Work

In this chapter, we highlight other works related to short text and invoice classification. Short-text classification is a broader area and some solutions may not be suited for invoice classification. In contrast, works aimed at invoice classification may not utilize short text processing techniques.

This chapter is organized as follows: the first section presents non-neural methodologies for short text processing. Section 3.2 will focus on neural-based short-text classification architectures. In section 3.3, we present recent work on invoice classification. We then summarize the approaches presented in this chapter based on a set of desirable characteristics for short-text and invoice classification. Finally, we point out the intended contributions of this work.

## 3.1 Non-neural Methods

Traditional methods rely on bag-of-words representation and matrix factorization to create a representation for text processing. The low word count on short text documents leads to low co-occurrence of terms across the document-term matrix, which invalidate matrix factorization methods.

Early works attempted to address this problem by expanding available information through auxiliary databases. Document expansion seeks to substitute the representation of short text for the representation of a set of related documents. In query-based expansion, these documents are returned by using short text as the input on a search engine [23][24]. The problem with document expansion is that it increases computational cost both on searching and processing a larger amount of data. This new data also introduces noise to the model.

Phan [56] proposed a framework for short text classification that used an external "universal dataset" to discover a set of hidden topics through Latent Semantic Analysis.

The discovered topics were used as a representation for short text. The framework was evaluated on two different tasks: domain disambiguation for web search results and disease classification of medical abstracts. The framework is simple enough to be adapted to different domains, with a "universal dataset" been utilized for multiple problems. Nonetheless, it is necessary to manually construct the dataset in a way that the hidden topics are well represented.

Alsmadi [57] tackled short text problem by utilizing a supervised weighting scheme to the document term matrix. Since the number of terms in short text representation is lower than in those of longer length, representation schemes should have a bigger focus on individual term strength. The models improved F-measure when compared to traditional weighting schemes. Alsmadi [15] points out that even though many models perform well on benchmark datasets, they struggle with real-world problems.

Wang [58] compared Naive Bayes(NB) models and SVM modes on varied topic classification and sentiment analysis. Based on the experiments, it was concluded that both NB and SVM were useful baselines modes, with models outperforming one another based on the task. Bi-grams were also shown to improve performance across multiple tasks.

## 3.2 Neural Based Methods

Neural based methods learn feature transformation on each of its layers. This allows non-neural methods to learn representations from higher dimensions feature inputs and also preserve sentence structure. Neural network generally use embedding vectors previoulsy trained on a auxiliary self-suppervised task as inpyt . Convolutional neural networks learn a new representation by passing filters through the input vector, caputuring the presence of sub-structures. Reccurent neural netwokrs process the input in vectors in sequence, creating a hidden state that is dependant on the order of inputs.

The architecture proposed by Kim [33] serves as the basis for most CNN based solutions. Zhang [59] utilized a 12-layer CNN to learn features from character embeddings. Character-based representation does not rely on pre-trained word embeddings and could be used in any language. Wang [9] expanded the model proposed by Kim [33] by utilizing concept expansion and character level features. The model utilized knowledge bases to return related concepts and included them in the text before the embedding layer. Knowledge bases included: YAGO, Probase, FreeBase, and DBpedia. A character-based CNN was used in parallel to the word concept CNN. Representations learned by both networks were concatenated before the final fully connected layer.

Naseem [10] proposed an expanded meta-embedding approach for sentiment analysis of short-text that combined features provided by word embeddings, part of speech tagging,

and sentiment lexicons. The resulting compound vector was fed to a Bi-LSTM with an attention network. The rationale behind the choice for an expanded meta-embedding is that language is a complex system and each vector provides only a limited understanding of the language. BiLSTM was used to address the semantic meaning of negation and conjunctions that are based on word order.

Chen et al [60] proposed a 2-D TF-IDF feature representation as input to CNN and LSTM models. These models were used to classify tweets based on verbal aggression. In the experiments, 2D Tf-IDF outperformed Word2vec embeddings trained on problem data. Due to the small dataset and shortness of each input, word embeddings trained on domain data failed to capture word relationships such as positive words (successful, happy), negative(worst, lose) and neutral (everyone). Regarding the architecture, while a single layer of convolution produced the best results, as the number of convolutions increased, the number of parameters dramatically decreased.

## 3.3 Invoice Classification

Electronic invoices are a special kind of short text. Invoices are particularly brief, being composed of only a few words, not forming a complete sentence. Words are often abbreviated and domain-specific vocabulary is abundant. Invoice classification techniques have ranged from traditional count-based methods to neural-based architectures. In 2017, chinese invoice data was made public for chinese researchers, which motivated research in the area. This leads to the prevalence of works dealing with the chinese invoice system.

Some works aimed to address data sparsity problem by utilizing hash trick for dimensionality reduction [4][5]. Yue [5] performed semantic expansion of features through external knowledge bases before using the hash trick for dimensionality reduction. Tang [6] utilized paragraph embedding to create a reduced representation and then applied K-NN classifier. Yu [7] utilized a parallel RNN-CNN architecture, with the resulting vectors being combined in a fully connected layer. Zhu [8] combined features selected through filtering with representation learned through the LSTM model.

Unlike most western languages, in which text is expressed through words with white spaces as separators, text in Chinese is expressed through characters without separators, with no clear boundary. Words are constructed based on the context. Most of the cited works used jieba[1] for word segmentation. Chinese invoice classification words leaned towards RNN based architectures in a way to mitigate error produced in the word segmentation step.

[1]available at: https://github.com/messense/jieba-rs

Chinese works aside, Paalman et al [61] worked on the reduction of feature space through 2-step clustering. The first step was to reduce the number of terms through filtering and then cluster the distributed semantic vector provided by different pre-trained word embeddings. This method was compared to traditional representation schemes and matrix factorization techniques. In the experiments, simple term frequency and TF-IDF normalization performed better than LDA and LSA.

## 3.4 Comparison of Methodologies

Traditional methods mainly address the data sparsity problem of short text. Traditional methods are based on filtering, semantic expansion, or dimensionality reduction through the hash trick. The problem with filtering is that there is information loss in a context where information is already poor. It is also vulnerable to typos, multi-word expressions, domain-specific vocabulary, and does not account for new vocabulary as time passes. The benefit of the hash trick over filtering is that no term, aside from stop words, is excluded from the representation. This guarantees that there will be a corresponding representation for every possible document. The downside is that there is no semantic meaning in the collisions that occur in terms that are represented in the same bucket. Nonetheless, the hash trick presents all the other downsides of filtering.

Semantic expansion can be done in conjunction with other methods and is based on increasing available information by leveraging external knowledge bases and thesauri. This increases processing cost and search-overhead. At the word-level, each word of the document will trigger a search for synonyms or related concepts. Communication with knowledge bases becomes the bottleneck of the system. The overhead in processing makes it unsuited for invoice processing due to the amount of invoice data. Furthermore, knowledge bases are language-specific, costly to build and maintain, and may not be available in languages other than English and Chinese.

Phan's approach [56] manages to circumvent the data sparsity problem without creating a processing overhead during training; instead, there is an increased effort at the preprocessing stage in creating the universal dataset. The "universal dataset" also requires domain expertise to correctly choose examples for each class.

Word embeddings serve two purposes in short-text processing. First, they provide a vector representation in fixed size for each word. This bounds the dimensionality of the representation based on embedding and sentence length. This contrasts with count-based representation, in which the number of dimensions depends on vocabulary size. The second benefit is that word embeddings capture semantic concepts between words based on the distributional hypothesis. This property allows for similar words to have similar

vectors. In a way, this functions similarly to semantic expansion, in the sense of synonyms expansion, without the drawback of processing overhead.

The limitation of word embeddings comes down to vocabulary coverage and word sense. Word embeddings cover the vocabulary present in the dataset they were trained on. During training, rare words are often dropped. Most architectures utilize pre-trained embedding as lookup tables to extract vector representations for each word. If the word is not covered by the embedding or if it is misspelled, the lookup table will fail to wield a representation and that vector will be randomly initialized. The second limitation of word embedding is that they capture the most common sense for each word based on the most frequent use of that word in the training data [50]. This may harm the representation of words with multiple senses.

These limitations of word embeddings are significant to invoice classification. Words in invoices are often misspelled and abbreviated. Also, taxpayers often mix words of multiple languages depending on the kind of product being reported. Finally, invoices possess little to no context in order to disambiguate word sense.

The literature indicates that a invoice classification architecture must the address the following issues:

- Character level representation: One of short-comings of word-base representation is that words that are not in the indexed word embeddings lack a corresponding embedding vector. This words are called out-of-vocabulary tokens, and are represented by random initiate vectors. Character level representation allows for representation of out of vocabulary words in the context of word embeddings due to new words being a combination of previously seen characters. This covers both misspellings and abbreviations, as the resulting representation of this words from the character embeddings would be more similar to the original word embedding than a randomly initiated vector.

- contextualized word vectors : In short-text classification, only few words are actually useful for classification. By using a convolutional layer with filters of different window size and a single word slide, it is possible to generate features for each word that take context in consideration. This is particularly useful for multi-word expressions.

- Sequence modeling: Word order may play a significant role in semantics, mostly in cases involving negation and conjunctions in the sentence. LSTM are also able to leverage more distant relationships then CNN.

- meta-word embedding: improves both vocabulary coverage and sense expression. Word representation learned by word embedding depends not only on the training

algorithm but also on the dataset in which it is trained. Invoice text differs in a great deal to natural language, so both word similarity and word sense capture by these embeddings may not reflect the characteristics of words present in the data. By combining multiple word embededings we allow the model to learn which is more useful for the task at hand. Pre-trained embeddings can also be combined with domain specific embedding

- Knowledge Base Independence : Due to the overhead in communication with external databases, knowledge base expansion may not be feasible for large scale datasets ,such as invoice data. Furthermore, these resources are expensive to build and maintain and may not be available in languages other than English and Chinese.

With this work we expect to provide the following contributions:

- A study on available machine learning classification models to assign the correct NCM code based solely on the product description.
- The architecture for a Intelligent system to assist tax auditors.
- A contextual framework that organizes the possibilities in invoice processing based on document aggregation and complexity, from product transaction to issuer processing.

# Chapter 4

# SCAN-NF and Invoice Processing

The goal of this chapter is to present our approach to invoice classification that consists of both a contextual framework to model the problem context and an architecture for a system to aid tax auditors, named SCAN-NF. This chapter is organized as follows: first, we present an explanation on the electronic invoice model is provided before a contextual framework for invoice processing is presented. This framework addresses the purpose of this field of research, the core concepts, research opportunities and how the present work fits the larger context. The last sections are dedicated to presenting the architecture of SCAN-NF, a system to aid tax auditors in identifying suspicious product transactions.

## 4.1 Electronic Invoice Documents

In the last 10 years, there has been an increased interest in electronically processing invoices due to both an improvement on the available computational resources and techniques, as the cost of processing large amounts of data have been decreasing year by year. The emergence of e-commerce as a regular practice has also contributed to the popularization of e-invoicing. While technical implementations may differ form location to location, the overall structure of this documents is the same. They will contain identification of both parties, information about the products and services contained in the transaction, and other general metadata such as date and type of transaction. Structuring this information may prove to be difficult depending on the degree of e-invoicing maturity. For mature landscapes, retrieving a semi-structured dataset may be as simple as providing a SQL query to a data-warehouse, in other cases this means utilizing computer-vision applications to extract information from a large amount of scanned physical documents.

Due to an availability of invoice data, this work will focus its on the Brazilian model. Nonetheless, our work could be useful for similar languages due to the similarity in invoice documents.

### 4.1.1 Brazilian Electronic Invoices

Brazil utilizes two types of electronic fiscal documents. The NF-e is the electronic fiscal document, created to substitute physical invoices, providing judicial validity to the transaction and real-time tracking for the tax office [62]. The NFC-e is a special case of the NF-e, issued by retailers for transactions with end-consumers. Some of the fields present in the NF-e are not mandatory in the NFC-e.

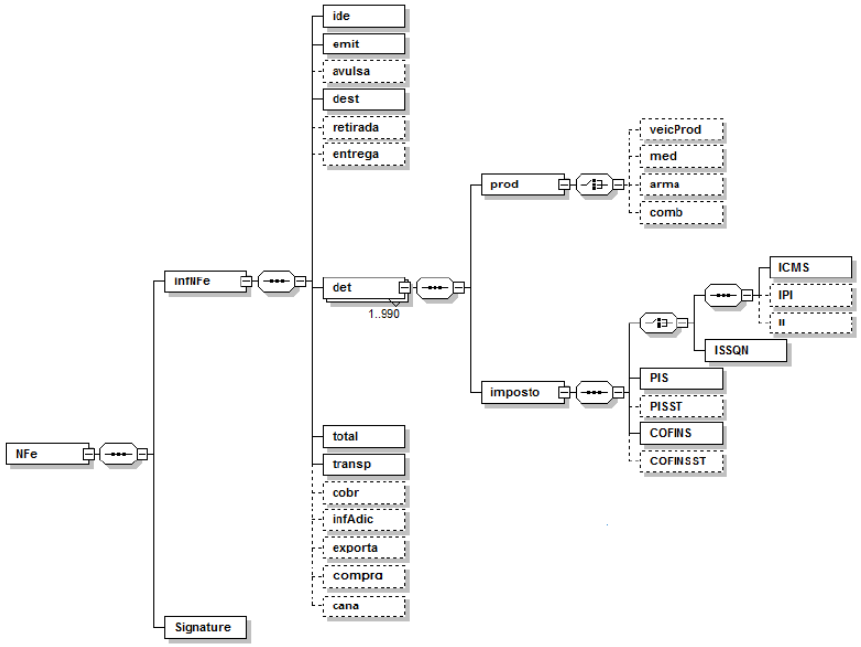



Figure 4.1: Diagram of the information present in the NF-e. Extracted from [62]

Figure 4.1 presents the nested structured of the NF-e XML file. It contains detailed information about invoice identification, issuer identification, recipient identification, product, transportation, and total values. In this work, we will focus on product information. Table 4.1 presents the information contained in product XML node.

Our analysis will focus on learning a feature vector for the product description field. To do so we will try to predict the NCM code. NCM is a regional standardized nomenclature for products adopted in Brazil, Argentina, Paraguay and Uruguay. It is used in all external commerce made by these countries. Once it has been known, the product is represented by the NCM code in all future processing.

There are validations rules for the NCM field in the NF-e manual[62]. The NCM field is obligatory and should contain all 8 digits, it should be a valid NCM code, and the NFC-e can only contain certain types of products. According to a specialist working with tax audition and the schedule published in the NF-e manual, the validation procedures

| Field | Description | Observations |
|---|---|---|
| prod | | Xml parent node |
| cPROD | Issuer Product Code | Product Id in the issuer system, not to be confused with NCM code |
| cEAN | Global trade item number | GTIN-8, GTIN-12, GTIN-13 or GTIN-14, blank if the there is no GTIN for the product |
| xPROD | Product description | |
| NCM | NCM code | Must be complete 8 digits, or '00' for entries that do not constitute products |
| NVE | NVE code | Details certain NCM |
| EXTIPI | exemption code | Some products may present different taxes from those with the same NCM due to tax exemption |
| CFOP | CFOP code | 4 digit code that describes the transaction |
| ucom | Comercial unit | E.g. kg, ml, L, un, etc |
| qCom | Comercial quantity | |
| vUnCom | Individual value | |
| vPROD | Total value of the product | |
| cEANTrib | GTIN of the taxable unit | |
| uTrib | Taxable unit | |
| qTrib | Taxable quantity | |
| vUnTrib | Unitaty tax value | |
| vFrete | Shipping value | |
| vSeg | Insurance value | |
| vDesc | Discount value | |
| vOutro | Complementary expenses | |
| indTot | Tag if the vProd sums to the total value of the NF-e | |

Table 4.1: Fields contained in the product node of the NF-e. Adapted from [62]

for the NF-e documents are currently implemented, while the same procedures are yet implemented for NFC-e documents. In the Federal District, NFC-e has only become mandatory as of late 2017. This results in data of poor quality, with lots of missing fields and no validation on whether or not the contained information is true.

The correct evaluation of the NCM is of great value in processing invoices. From the NCM code, it is possible to retrieve the correct tax rate and better monitor the circulation of products. This, in turn, can aid the decision making of the legislative and executive branches. We can also cross the information contained in different fields of the invoice to check if the values reported seem to be valid. When dealing with invoice processing, we conceptualize different degrees of abstraction. From the bottom up, we can address problems based on product transaction, invoice transaction and invoice issuers. At product level we are dealing with the individual products being referred into each invoice. The goal at this level is how to represent products and how address missing information. Stepping up, we can deal with invoices. Each invoice invoice is a collection of product transactions between two parties at a given date with additional meta-data. When dealing with invoice data, new questions can be addressed such as product co-occurrence, transaction frequency and invoice data consistency. At the upper level, we can investigate business behavior by the flux of products described in the invoices. For fraud analysis, we can conceptualize an analysis that that initiates from product analysis up to issuer. In this indicator suspicious product description act as the basis. From suspicious products we can tag suspicious invoices. Based on sum of values present in suspicious invoices, we can decide if it is economically viable to audit issuers.

This works addresses invoice processing problem at product level for two reasons. The first reason is that before higher level processing can be done, handling how to represent product transaction is crucial. The second one is due to security reasons. When dealing with a large amount of invoices from a particular region, it is possible, with clever processing, to recreate economical behavior of key businesses. This kind of work would require closer partnerships with the treasury office due to legal and ethical reasons.

Our goal is to use text description present in each invoice to identify the product in that transaction. For this work, we will evaluate different text classification methods on NCM classification. NCM codes are used to represent products in many transaction processing steps. If the system learn how to correctly assign NCM code based on product description, that means the system was able to extract a useful representation from product description. Assigning NCM code is also useful for Tax auditors, since there is no current audition of the NCM being reported. The reported code could be missing, out of format or be wrongfully reported, either intentionally or not.

Our second goal is to verify if it is possible to leverage information presented in NF-e data to address NFC-e data. Since there is an overlap in the products described by both products, we raise the question if models pre-trained on NF-e data can be used on NFC-e data, or if patterns learn by models trained on NF-e data could be leveraged for NFC-e data. NF-e is historically issued by larger business with more mature audition processes, which leads to better data quality. If we could perform transfer learning from NF-e focused models to NFC-e model, this means that NFC-e models could be built from already existing NF-e models.

## 4.2 Contextual Framework

In this section, we present a contextual framework to understand the landscape of invoice processing. The framework is organized in a layered structure, with each layer representing a sequential step in invoice processing. Figure 4.2 presents a visual representation of the proposed framework. At the base level, there is the data structuring layer, followed by different invoice abstraction layers.

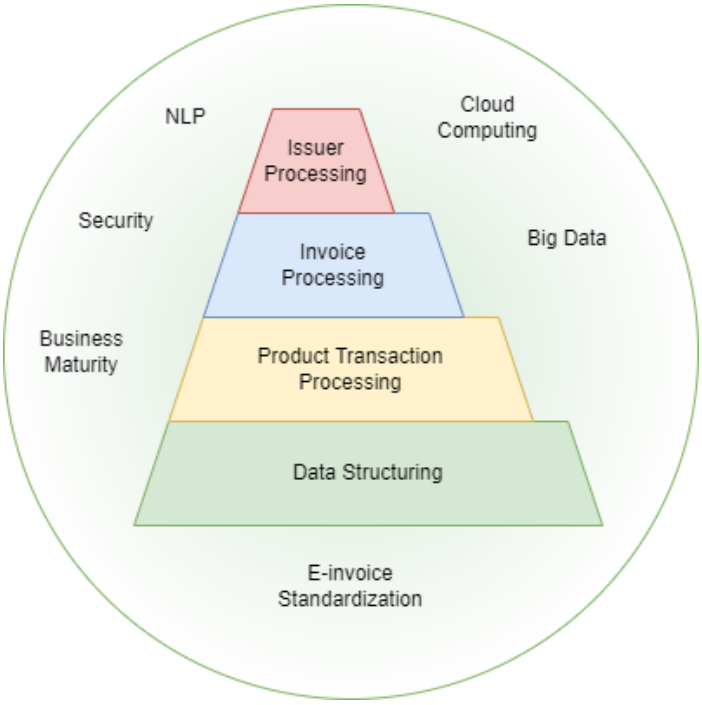



Figure 4.2: E-invoice Processing Framework

Although electronic invoices have become more and more popular in recent years, in many cases, useful documents only exist in physical forms or user-oriented digital files, such as document pictures and PDFs. Before processing any meaningful information, we need to extract data from these documents and store it in some semi-structured mode.

Related works have shown that computer vision solutions are useful for extracting useful information from physical documents directly [63][64][65] [66][64]. These methods can greatly reduce the costs and workload for generating invoice data sets. This task is especially important in auditing, because it is necessary to cross the information reported in invoices with sales records in other systems.

The remaining steps in our framework relate to different levels of abstraction that can be applied to invoice modeling. These steps include product transaction processing, invoice processing and issuer processing. Each level serves as the stepping stone for the next. Product transaction is the first layer of processing, representing each individual product or service transaction represented on every invoice in the data. At this level, we are interested in extracting granular information such as product description, product price, due taxes as well as other task-oriented attributes. The main form of input at this level is the product description. Our work is situated at this level, as we treat product description as a short-text classification problem to predict the correct product code for each transaction. This exemplifies the main concern at this stage: we are interested in creating a good representation for each product transaction in order to produce the input for later tasks. It is much easier to analyze products transactions from a standardized product taxonomy than processing text descriptions[67].

At the invoice processing level, individual product transactions are aggregated and used to represent each invoice in conjunction to other meta-data. It is possible to track the relationship between multiple products in the same invoice. For example, Paalman [61] utilized two-step clustering to track fraudulent invoices. Auto-encoders have also been employed in fraud detection by measuring the distance between the reported text and the expected text produced by the model [68]. At this level, we can also model consumer behavior by utilizing association rules based on common product transactions. Another example is the usage of invoices issued by healthcare centers to extract association rules between commonly used medication [69].

At the higher level of abstraction, the behavior of parties involved in transactions is taken into account. One approach is to utilize previously known troubled issuers as a flag in processing invoices. An example of this kind of procedure is Chang's work [3], in which information about companies involved in violations is used to select and mark invoices to create an alarm system for safe edible oil. Another way to include issuer analysis in invoice processing is through graph analysis. It would be possible to model an oriented graph, each node representing an issuer with invoices being used to create the links between issuers. From this structure, it would be possible to look for communities, cycles, and other graph-oriented sub-structures and correlate them to real-world issues. At the time of this work, we have not been able to find works that model invoice processing utilizing

graphs. We hope to address this tackle this problem in the future.

### 4.2.1 Larger Context

Invoice processing is also related to other concerns that are not directly related to extracting information from invoice documents. Due to a large amount of data, invoice-based systems require big data architecture [70]. This may lead to solutions in distributed computing paradigm as storing and processing are more feasible in clusters than in single machines. The adoption of e-invoicing from the get-go is also a key factor, as it streamlines the data structuring layer, doing away with the need of using expensive image processing techniques to create digital representations of invoices. A maturity model for e-invoicing from the business perspective was provided by [71].

## 4.3 Architecture of SCAN-NF

We present an overview of the architecture of the SCAN-NF system and inner model in Figure 3. The system's goal is to feed additional information for tax auditors by product code to each product transaction based on the product description. The labeled transaction is then used as inputs for other analyses by Tax Auditors and Specialists. The system works in two phases: a training phase and a prediction phase. During the training phase, the system is fed audited data from tax office server to train a supervised model. Two models are trained, one for the classification of NF-e Documents and another for NFC-e Documents. After training, these models are used on new data during the prediction phase. The system works as follows: Data is extracted from the tax office server (Label 1 in Figure 3). At this point, data provided in by the server may not be in a format compatible with the classification model. Product description and corresponding NCM code for each product in each invoice are then extracted (Label 2 in Figure 3). Text is then cleaned from irregularities (Label 3 in Figure 3). A training dataset is constructed by balancing target classes samples and dropping duplicates (Label 4 in Figure 3). At this point data is already modified to fit the model. The training set is then fed to a model that learns to classify product descriptions (Label 5 in Figure 3). Outputs at the training phase of the system are used to validate models before being put into production (Label 6 in Figure 3). During the Prediction Phase, trained models are utilized to classify new data. These datasets may be composed of invoices issued by a suspected party or a large, broad dataset used for exploratory analysis (Label 7 in Figure 3). Models trained in the training phase are then employed for the task at hand (Label 8 in Figure 3). The final output of the model is the classified set of products inputs (Label 9 in Figure 3). This set

of classified product transactions is then used in manual auditing by tax auditors (Label 10 in Figure 3).

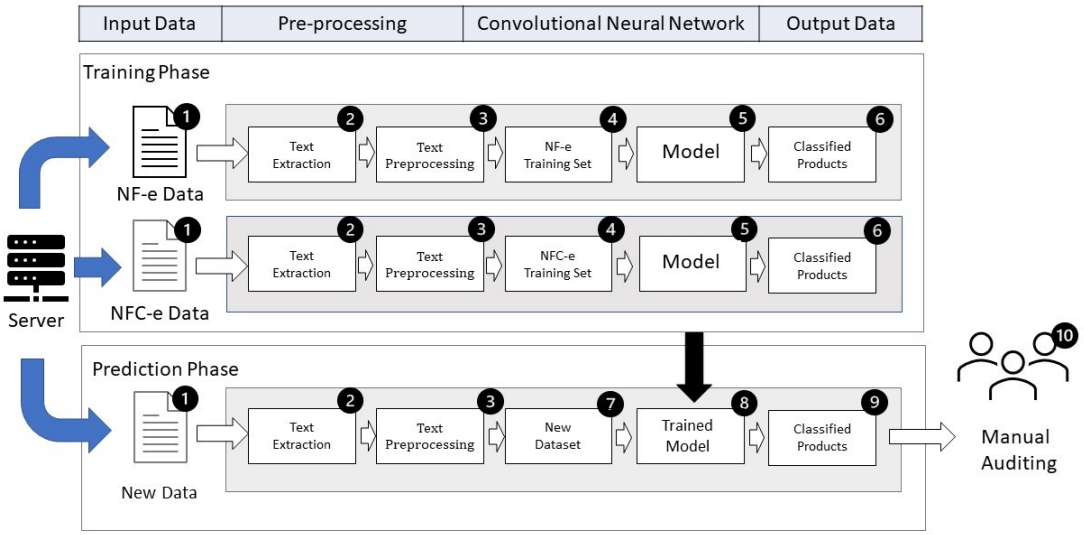



Figure 4.3: Architecture of SCAN-NF. Extracted from [1].

The system is intended to aid tax auditors in auditioning invoices issued by already suspicious parties to pinpoint inconsistencies and irregularities. Currently, NFC-e documents are not audited due to the amount of data, a more significant number of issuers, and the nature of the data. Our solution helps auditors pinpoint inconsistencies in documents reported by an already suspicious party and allows for the automatic processing of more data. We hope that this solution will improve the productivity of tax auditors regarding NF-e processing and be the first step towards NFC-e processing.

## 4.4 Case Study of Brazilian E-Invoices

In order to validate SCAN-NF architecture, we aim to address the following questions:

- Is it possible to utilize product description to classify the NCM code of Brazilian Electronic invoices by using State of the Art Text Classification methods?
- How does this methods compare to traditional Machine Learning Methods?
- Is it possible to train models work across different types of electronic invoice documents?

To answer this questions, we conducted a case study based on real NFC-e and NF-e documents from the the state treasury office (SEFAZ). Data were separated into training and test sets, and different models were trained. Models were validated through cross-validation. Hyper-parameter optimization was conducted based on the average performance through all folders of cross-validations.

### 4.4.1 Dataset

In our experiments, we utilized data provided by the estate tax office of SEFAZ. Data provided included both NFC-e and NF-e documents. NF-e data consisted of a small curated dataset that was previously used in manual analysis of product codes and description on various products. This dataset contained information on the description and NCM code. NFC-e data consisted of a larger dataset of products from multiple sectors, extracted from a particular month over the transactions conducted by retail supermarkets.

Due the lower number of rows in the NF-e Dataset, we established a treshold based on the number of unique rows of each class in the NF-e dataset, due its smaller size. We selected all classes with a total of distinct product description over 2000 rows. In total 18 classes were selected from NF-e dataset to be used in the experiments. The same 18 classes were selected to create the experiment NFC-e Dataset. Our NFC-e dataset is not labeled by tax auditors. From previous studies, tax auditors estimate that 5% of issuers miss-classify products in NF-e documents. We assume that issuers behave similarly with the NFC-e. Otherwise, if the majority of issuers presented fraudulent behavior, the task of auditing would become trivial as auditors would have to simply pick issuers with sufficient transactions to be cost effective to audit.

These datasets were kept separate and balanced individually. Due to disparity in market share, preserving product frequency would bias the models toward larger issuers and the most popular products. This could lead models to better classify invoices from large companies or learn their representation as to the norm. Duplicate product descriptions for each target class were drooped, even though they represented different transactions. While there is a significant vocabulary overlap between NF-e and NFC-e documents regarding NF-e data, NFC-e presents a much more vast vocabulary. Table 4.2 presents detailed information on the number of samples used in the experiment.

## 4.5 Discussion and Review

This chapter presented the modelling of the invoice classification problem as an short-text classification problem. The presented framework establishes different levels of abstraction for invoice processing. Each layer serves as the basis for the next layer. This work

Table 4.2: Number of samples and datasets used in experiments. Extracted from [1]

| | NF-E | NFC-E |
|---|---|---|
| Number of raw product samples | 198882 | 99637515 |
| Number of samples in balanced dataset | 36234 | 49536 |
| Number of balanced classes | 18 | 18 |
| Vocabulary Size | 3646 | 15312 |
| Shared Terms | 2342 | |

addresses invoice classification at the product transaction layer by presenting SCAN-NF, a system to aid tax auditors identifying suspicious transactions by classifying product transaction NCM code based on product descriptions. From the NCM code it is possible to identify the correct taxation to be applied to each product. The output of this systems could be used as input for more complex systems in order to identify fraudulent behavior. Both the research questions and the study case used to address this questions are outlined. In the next chapter, the methodology and experimental setup used in the study case is presented.

# Chapter 5

# Experiments and Methods

Chapter 5 presents details about the methodology and implementation of experiments aimed at answering the research questions presented in the last chapters. The questions were: (1) Would it be possible to assign NCM code for product transactions based solely on the product-description field ? (2) How dos the different machine learning models compare to one another on this task? (3) Could models trained on one type document perform well on the other type. Answering these questions allows us to validate SCAN-NF architecture and also allows for other applications focused on aiding tax auditors and public decision-makers. Furthermore, by comparing results between the chosen classifier architectures, we can validate if invoice product description follows the same paradigm as other short-text processing.

This chapter is organized as follows: In section 5.1, an explanation on experiments, baseline models and metrics employed is given. A Flowchart of the experiment process is presented and explained in detail. Section 5.2 and 5.3 present training details for SVM and CNN architectures respectively.

## 5.1 Experimental Setup

As mentioned previously, two balanced datasets were constructed for both NF-e and NFC-e invoices. Data is split between training and test sets with a ratio of 80 to 20. Parameter tunning was done using 5-fold cross validation. We train different models on each dataset to compare metrics between document types. Hyper-parameter optimization was conducted based on the average performance through all folders of cross-validations. The final result for each model is taken by averaging the result of several configurations trained independently on the best parameters. Results are then analysed and compared to a naive classifier that that guesses at random. This naive classifier serves as a baseline

for comparison and put in to perspective the metrics achieved by other models. Code related to these experiments are publicly available [1].

To address whether or not it is possible to utilize invoice data to classify the other type of invoice, we take the previously trained models and evaluate them on the dataset composed of the other document type.

### Baseline Models

Based on the literature and previous insights, SVM and CNN-based models were investigate as possible candidates to be used as the classification engine in SCAN-NF. SVM serves as a baseline model and a example of traditional frequency based classification. There is no consensus on the related literature whether or not frequency based model should perform well on short text classification. Works based on short text classification indicate that this type of model should struggle due to document-term matrix being particularly sparse and the lack of sentence structure modelling present in other architectures. Nonetheless, related work on invoice classification utilized frequency based model with good results. SVM have also been cited as a reliable baseline model for both topic classification and sentiment analysis.

In contrast with frequency methods, CNN and RNN based architectures were said to better preserve sentence structure information. This is particularly important in short-text, since it is expected for every input to have less information than larger documents. Related work on CNN based short text classification seemed take the architecture proposed by Kim as the base for the classification model and iterate over it with some form of knowledge expansion. While CNN preserve local sentence structure based on the words taken by each filter, there is no context representation.

### Metrics

We evaluate models based on the following metrics: accuracy, precision, recall, and F1-score. Metrics are calculated based on True Positives, True Negatives, False Negatives, and False Positives.

Accuracy is given by the rate of correct predictions over all predictions, it can be expressed as:

$$(TP + TN)/(TP + TN + FP + FN) \tag{5.1}$$

Top k Accuracy represents how often the correct answer will be in the top k outputs of the model. Accuracy is useful for getting an overall idea of model performance. In

[1] available at *https://github.com/diegokieck/mestrado*

unbalanced datasets, recall and precision can paint a better picture of how the model behaves.

The recall represents the recovery rate of positive samples and is given by:

$$TP/(TP + FN) \tag{5.2}$$

Precision evaluates the correct set of retrieved samples and is given by:

$$TP/(TP + FP) \tag{5.3}$$

We utilize the F1-score, the harmonic mean of precision and recall, to get a balanced assessment of model performance on imbalanced classification.

### 5.1.1 Experiment

Figure 5.1 presents a flowchart of the machine learning pipeline. After data was provided by the state treasure office, research opportunities and solutions were identified through exploratory data analysis. At this stage, prior experiences in the treasure office with private consultants led to the belief that it wouldn't be possible to utilize text descriptions to classify invoice documents.

The idea that product description text is too poor to be used may have come due to challenges in establishing rule-based algorithms for product description classification. These studies were unavailable for the development of this work. While these rules could be extracted from the NCM taxonomy structure, setting up and maintaining such rules would be infeasible.

The problem was framed as a short text classification problem. Product description fulfills all the criteria for short text: brief text with low term count; text with poor grammar; the presence of meta-data, and individual author contribution being small. From the target variable, NCM, it is possible to assign the correct taxation type based on business rules.

Item description duplicates were dropped to not bias the study towards large-scale businesses and frequently sold products. The selection of which classes should be used in the case study was based on the constraints of the provided NF-e dataset. Classes had to present at least 2000 distinc product descriptions. 18 classes met this criteria in the NF-e dataset. The NFC-e dataset was created by sampling distinct transactions for the same 18 classes. Data preprocessing consisted of lower casing, removing punctuation and accentuation, and tagging numbers and metrics.

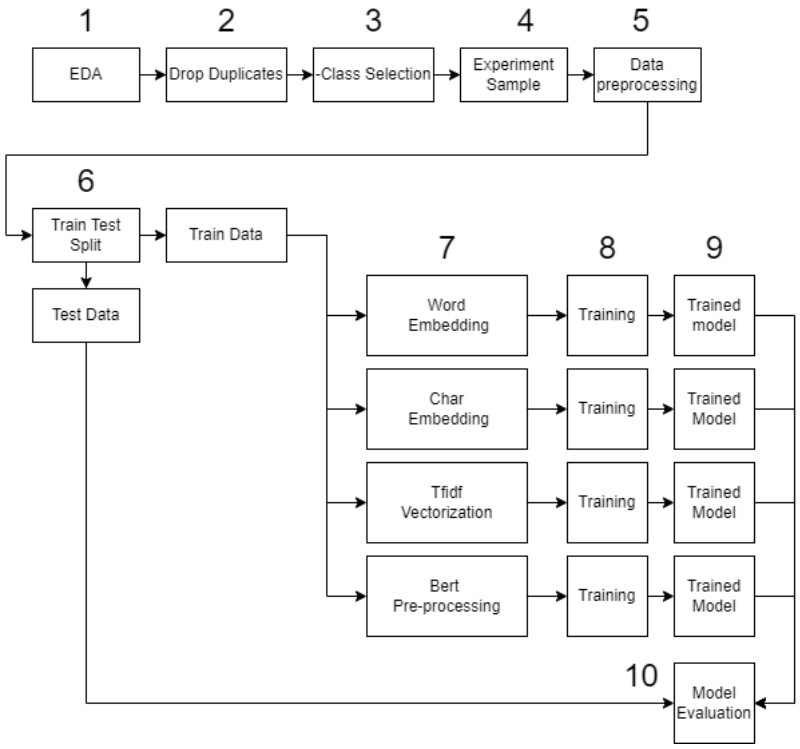



Figure 5.1: Experiment Flowchart

Train test split was done at an 80-20 ratio, with test data being set aside for the final model evaluation. Training Data needed to be prepared differently based on each model. Hyper-parameter optimization was conducted both on model and data preparation parameters. Records of each parameter trial were kept and used to select the final pipeline for each model. The average of the metrics over the set of ten experiment runs constructed the final results. For each experiment run, models were initialized, trained, and evaluated.

## 5.2 SVM

The Support Vector Machine implementation used was provided by the *Sklearn* library. For the initial evaluation, the model was presented with different kernel functions and different parameters for the selection of features. Text was vectorized through Sklearn TF-IDF vectorizer. Different parameters for maximal an minimal document frequency was used, as well as n_gram combinations from 1 up to 3. Maximal and minimal document frequency establish thresholds for the frequency for terms to be added to the vocabulary.If a term is present in more documents than the maximal document frequency threshold it is not added to the vocabulary, the same is true if the term is present in less documents than the minimal frequency. N-grams define the length of terms combinations to be added

to the vocabulary. N gram is defined as a range that includes all n-grams between the limits.

| Parameter | NF-e Dataset Values | NFC-e Dataset Values |
|---|---|---|
| Kernel type | [**'linear'**, 'poly', 'rbf', 'sigmoid']) | [**'linear'**, 'poly', 'rbf', 'sigmoid']) |
| max_df | [1.0, **0.5**, 0.75] | [1.0, 0.5, 0.75] |
| min_df | [**1**,0.1, 0.01, 0.001,0.0001 ] | [**1**,0.1, 0.01, 0.001,0.0001 ] |
| n_gram | [(1,1), **(1,2)**, (1,3)] | [(1,1), (1,2), **(1,3)**] |

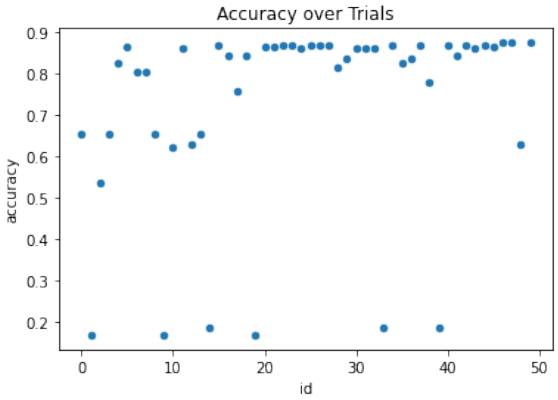
Table 5.1: Hyper-parameters for SVM training

Table 5.1 presents the hyper-parameters used in optimization, with the best results in bold. Figure ref presents results across trials. The best parameter over all trials was used to train the final model. Hypeopt package was used to search for the best parameters. Hyperopt applies Bayesian inference to predict most likely best parameter in exploratory manner, converging to the best parameters over iterations.

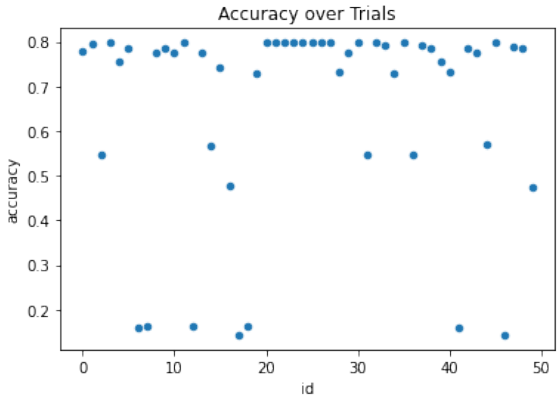
Figures 5.2a and 5.2b present Accuracy over all trials for each dataset over iterations of Hyperopt trials. We can see that models peaked at around 0.9 accuracy-score on the NF-e dataset and 0.8 accuracy-score on the NFC-e database. We can see three clusters with different ranges of accuracy. On the bottom figures 5.2c and 5.2d, we have trial results ordered by training time, in which the y axis represents accuracy metric and x axis represents time to train that configuration. By looking at the left bottom side of theses two graphics we can see that the worst performing configurations had the least training time. This is explained by their choice of vocabulary. A lower vocabulary size generates a lower number of dimensions in TF-IDF, witch facilitates SVM convergences due to a lower number of kernel transformations. However, less information is presented to the model. In contrast, at the right side of the graphics we have configurations that took significantly more time to train but performed no better than faster configurations. This indicates that there is a limit to the relevancy of information that is useful for the model, including more terms or high n-grams to the vocabulary only slows the model and does not provide better.

## 5.3 CNN architectures

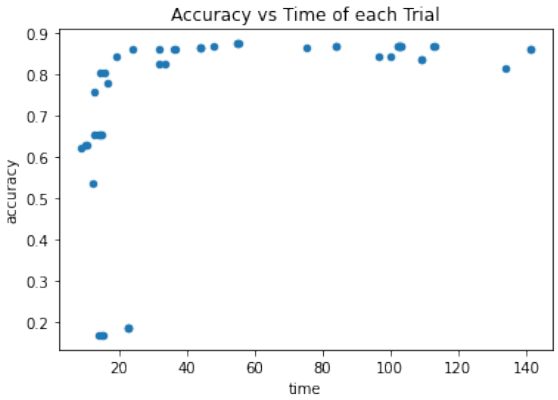
The Keras API for the TensorFlow library was used to implement the CNN models. The architecture consists of three channels with different kernel sizes concatenated and fed to a dense layer. Hyperopt optimized the mean validation accuracy over 10 folds. Optimized parameters consist of: the number of filters in each channel, the size of the dense layer, the dropout rate, the size of the second dense layer, and the optimizer.



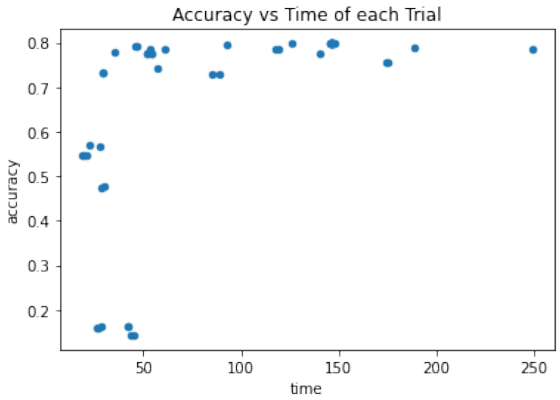

(a) Accuracy over each Trial for the NF-e dataset.




(b) Accuracy over each Trial for the NFC-e dataset.




(c) Trials sorted by training Time for NF-e dataset.




(d) Trials sorted by training Time for NF-e dataset.

Figure 5.2: Results of Hyper-parameter Tunning of SVM model

Figure 5.3 presents the base CNN architecture used for CNN models. The input sentence is fed to a embedding layer. In the embedding layer each token is replaced by the corresponding word embedding. After that, input is reshaped to fit the next layers. The input is fed to 3 different channels that will apply 1D convolutional kernels followed by max pooling. Each kernel functions as a filter that detects a relevant substructure in the sentence. Max pooling layers verify if the learned kernels were activated anywhere on the input sentence and pass on the information to the next layer. At this point each feature in the feature vector corresponds to the output of a learned filter applied to the sentence. The feature vector is flattened and passed through a dropout layer before being fed to a fully connected layer that will work as the classifier.

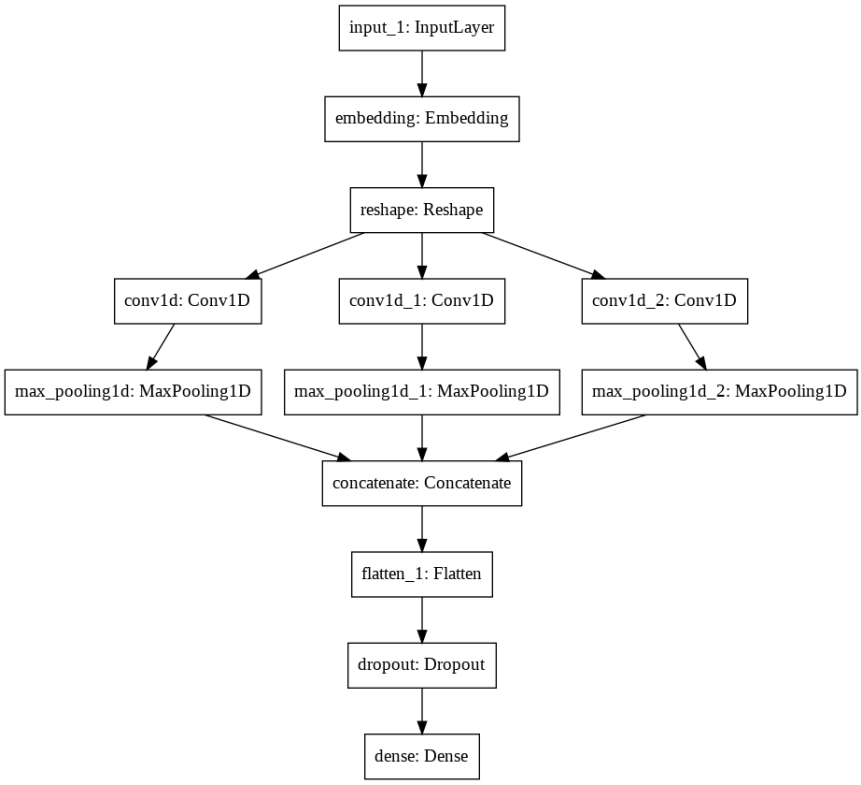



Figure 5.3: Base CNN Architecture

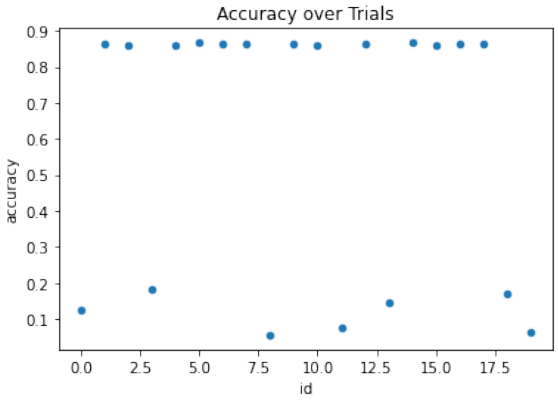
### 5.3.1 Word-CNN

The Word-CNN architecture splits the input sentence based on white space. Each word is substituted by a word vector with random values. 1-D convolutions go through the resulting sentence matrix computing filters of sizes 3,5, and 7.

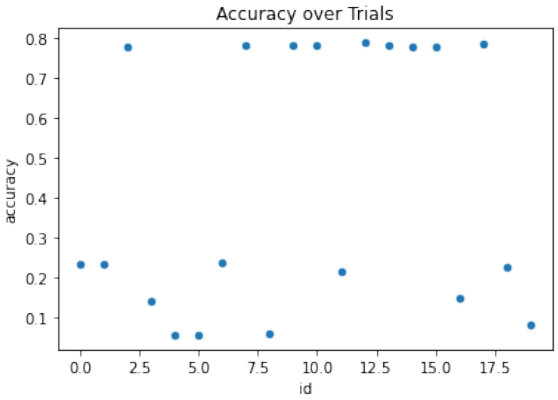
Figure 5.4 presents trial results for hyper-parameter optimization for the word-based CNN architecture over Hyperopt trials. We can see in Figure 5.4a and 5.4b that there is a lack of intermediate results in both datasets. Models either performed very similarly, around 0.85 accuracy score on the NF-e dataset and 0.8 accuracy score on the NFC-e dataset, or failed to learn, with accuracy lower than 0.2 accuracy score.

When visualizing accuracy over training time for the models in Figures 5.4c and 5.4d, we can see that models with a good performance, top left, trained faster than those that performed poorly, bottom right. Models may be too complex for the given task. Table 5.2 presents the hyper-parameters used during training. The best results are presented in bold.

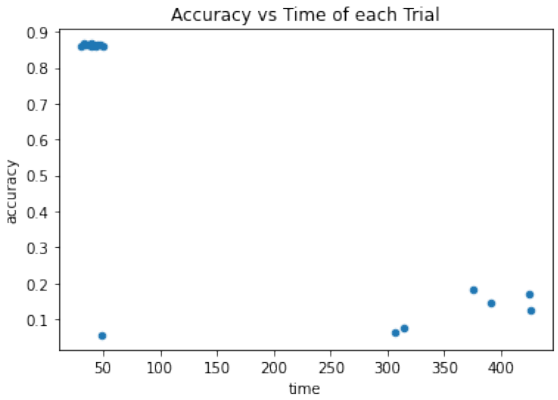
In Figure 5.5, sub-figures 5.5a and 5.5b present the training history and loss function for a model trained using the best hyper-parameter over the trials on the NF-e dataset and NFC-e dataset respectively. Accuracy on the NF-e dataset peaked at around 85% accuracy and 80% at the NFC-e dataset. From the graphs, we can see that the models



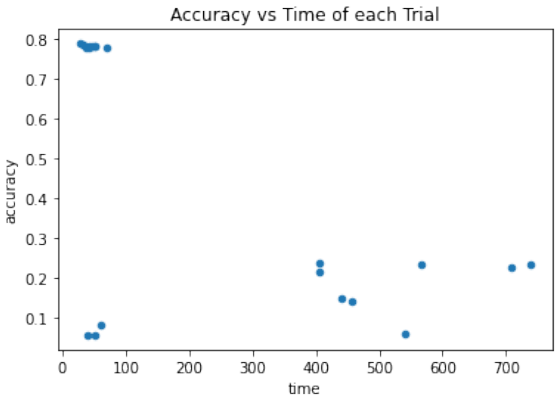

(a) Accuracy over trials results for the NF-e dataset.




(b) Accuracy over trials results for the NFC-e dataset.



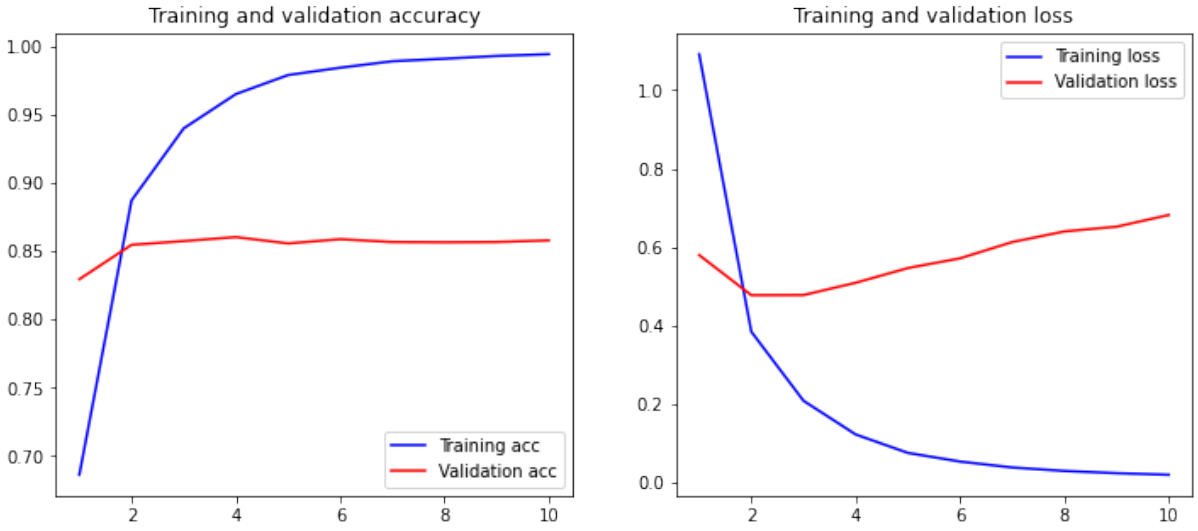

(c) Accuracy over time spent for each trial in NF-e dataset.




(d) Accuracy over time spent for each trial in NFC-e dataset.

Figure 5.4: Results of Hyper-parameter Tunning of Word CNN model

are overfitting from the second epoch onwards and may be too complex for the given task. We can see by comparing the curves of the training set and the validation set, loss on the training set continues to drop after the second epoch and starts to rise on the validation set. This indicates that the model is losing its ability to generalize its output. Either the architecture is too complex for the task or the amount of data for a convolutional modes was too small. This conflicts with the idea that short text classification requires specialized architectures. This raises the question if simpler models are more suited to the task.




(a) Training History of Word-Based CNN model on NF-e dataset.


(b) Training History of Word-Based CNN model on NF-e dataset.

Figure 5.5: History of Word-Based CNN model.

| Parameter | Values For NF-e Dataset | Values for the NFC-e dataset |
|---|---|---|
| Number of Filters on 1D Convolution #1 | {0,**300**, 600, 900, 1800} | {0,300, **600**, 900, 1800} |
| Number of Filters on 1D Convolution #2 | {0,**300**, 600, 900, 1800} | {0,**300**, 600, 900, 1800} |
| Number of Filters on 1D Convolution #3 | {0,**300**, 600, 900, 1800} | {0,**300**, 600, 900, 1800} |
| #1 Dense Layer | [**100**, 300, 600, 1000] | [100, 300, **600**, 1000] |
| #2 Dense Layer | [**0**, 100, 300, 600] | [**0**,100, 300, 600] |
| Dropout Rate | [0.0, **0.38**, 0.5] | [0.0,**0.27**, 0.5] |
| Optimizer | ['Adam', 'Adagrad', 'Adadelta', **'Nadam'**] | [**'Adam'**, 'Adagrad', 'Adadelta', 'Nadam'] |

Table 5.2: Hyper-Parameters for the Word-based CNN models trained on each dataset. Final parameters are presented in bold

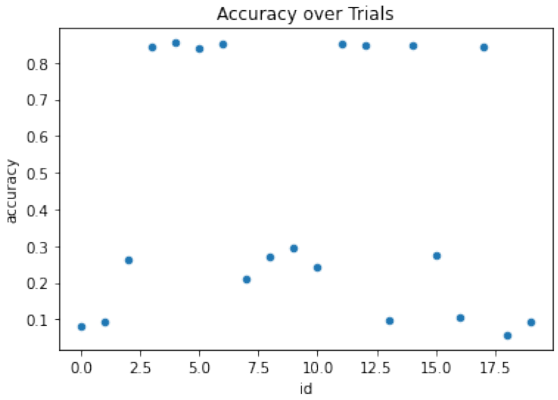
### 5.3.2 Char-CNN

In the Char-CNN architecture, tokens are generated from individual characters and assigned random initiated vectors. These vectors are fed to channels of different kernel sizes. Hyper-parameter optimization is aimed at finding the best size of neural network layers and whether or not the model could benefit from an extra dense layer.

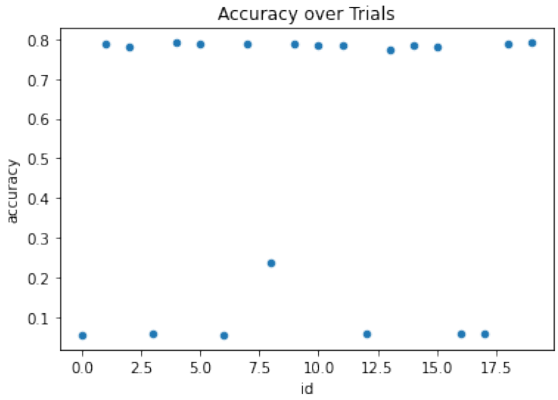
| Parameter | Values For NF-e Dataset | Values for the NFC-e dataset |
|---|---|---|
| Number of Filters on 1D Convolution #1 | {0,300, 600, **900**, 1800} | {0,300, **600**, 900, 1800} |
| Number of Filters on 1D Convolution #2 | {0,300, 600, **900**, 1800} | {0,300, 600, 900, **1800**} |
| Number of Filters on 1D Convolution #3 | {0,300, **600**, 900, 1800} | {0,300, 600, 900, **1800**} |
| #1 Dense Layer | [100, **300**, 600, 1000] | [100, 300, **600**, 1000] |
| #2 Dense Layer | [**0**, 100, 300, 600] | [**0**,100, 300, 600] |
| Dropout Rate | [0.0,**0.13**, 0.5] | [0.0,**0.37** , 0.5] |
| Optimizer | ['Adam', 'Adagrad', 'Adadelta', **'Nadam'**] | ['Adam', 'Adagrad', 'Adadelta', **'Nadam**'] |

Table 5.3: Hyper-Parameters for the char-based CNN models trained on each dataset. Final parameters are presented in bold

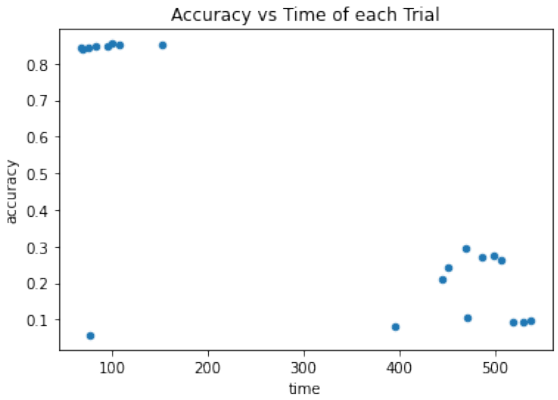
Figure 5.6 presents results metrics over Hyperopt trials. Similarly to the word-based CNN, we can see the models performed better on the NF-e dataset than on the NFC-e



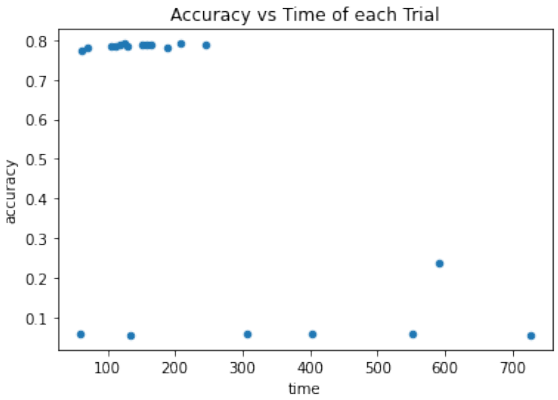

(a) Results of the Character-Based Model on NF-e Dataset




(b) Results of the Character-Based Model on NF-e Dataset




(c) Results on NF-e Dataset sorted by Training Time

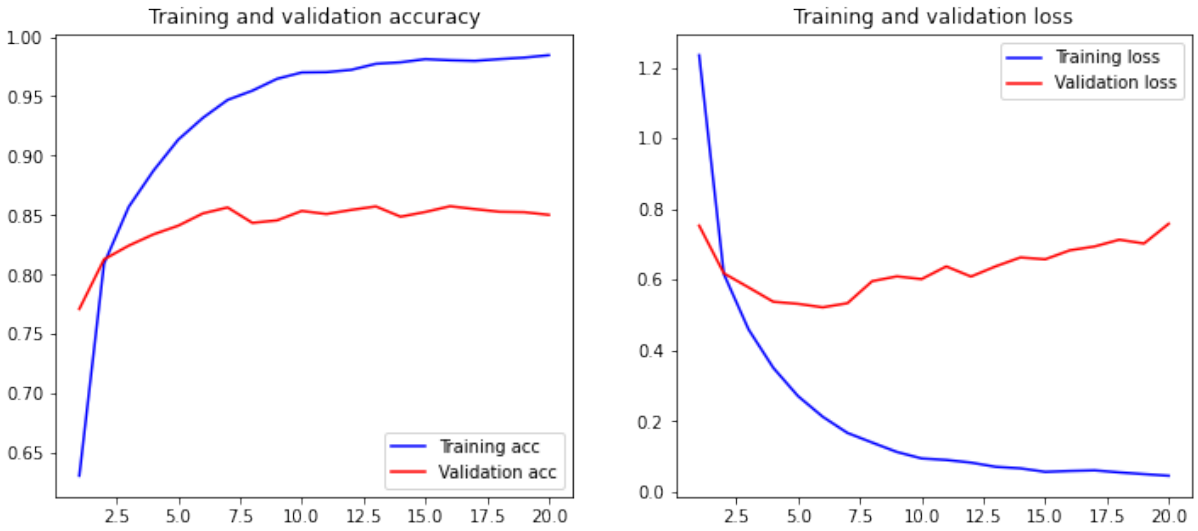



(d) Results of the Character-Based Model on NF-e Dataset

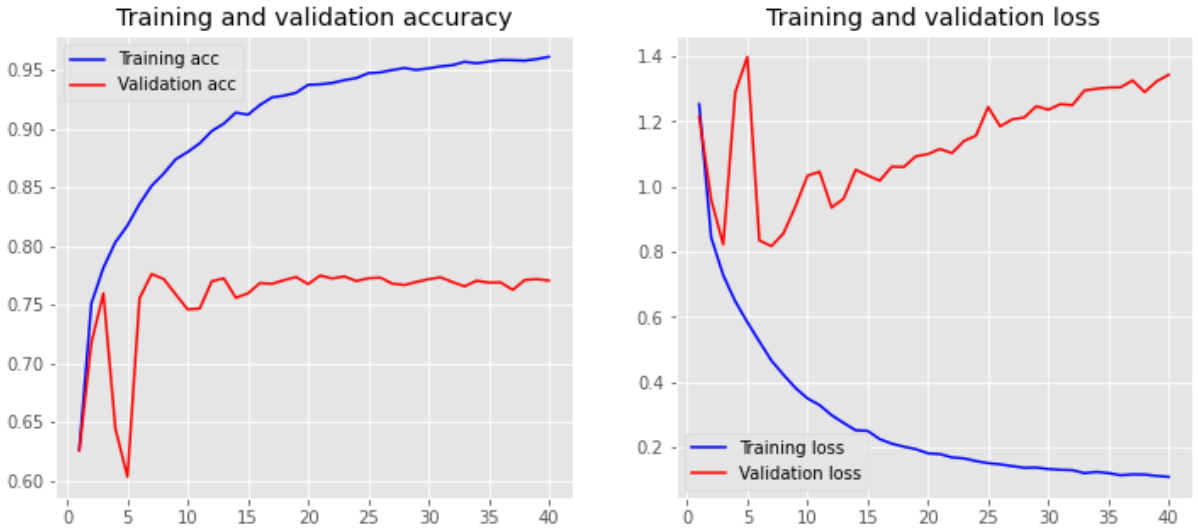
Figure 5.6: Results on NFC-e Dataset sorted by Training Time

database, with around 85% and 80% accuracy respectively. Similarly, when sorting results based on training time, configurations with higher accuracy had lower training time than configurations with higher training time. One possible cause for this phenomenon again is that increasing the model complexity had a detrimental effect on performance, increasing training time and lowering accuracy. Hyper-parameter values are shown in table 5.3.

In Figure 5.7, sub-figures 5.7a and 5.7b present the training history and loss function for a model trained using the best hyper-parameter over the trials on the NF-e dataset and NFC-e dataset respectively. Accuracy on the NF-e dataset peaked at around 85% accuracy and 80% at the NFC-e dataset. Similarly to what we have seen in the Word-Based CNN, models are over-fitting from the second epoch on-wards and may be too complex for the given task.




(a) Training History of the Character-Based CNN Model on NF-e Dataset.




(b) Training History of the Character-Based CNN Model on NFC-e Dataset.

Figure 5.7: Training History of the Character-Based CNN Model.

## 5.4 Discussion and Review

This chapter presented the methodology and experimental setup used to train the machine learning models that could power SCAN-NF. Models were trained on two different datasets, each representing one type of brazilian invoice document, NF-e and NFC-e. The chosen classification models were SVM classifier, Word-based CNN and Character-Based CNN. These models were chosen based on related literature. For each type of model, hyper-parameter optimization was conducted on both datasets separately. Convolutional models easily overfitted on both datasets from the second epoch onwards. During model optimization, models tended to either perform very well or very badly, with large gaps in performance between the best hyper-parameters and the worst. Models that took the longest to train failed to learn relevant patterns. The next chapter will present experimental test result of models trained on the best hyperparametes found in optimization.

# Chapter 6

# Case Study on Brazilian Invoice Data

This chapter presents results of experiment conducted in the study case using real world invoice data provided by the state treasury office. Experiments were conducted as described in the previous chapter. Section 6.1 addresses the question if its possible to identify the NCM code for a product based solely on product-description field. This is done by comparing model performance to a dummy baseline. In section 6.2, comparison between different approaches is further detailed. Section 6.3 presents the results of models trained on one type of document and applied in the other. The chapter concludes with an review and discussion of findings.

## 6.1 Classification of Invoices Based on Short Text Description

In this section, we explore experimental results. Table 6.1 presents the results of all models on both datasets together with a baseline for a naive classifier. Models had to assign one of 18 possible labels to each product, representing the correct NCM code. The naive classifier represents a model that does not look at the data and simply randomly predicts a target label. Results for each model are calculated based on the mean average of 10 runs, in which the model trains on the entire training dataset. Metrics are taken based on results of the test set.

This experiment aimed to address several questions. The first one was if it was possible to create a classifier that correctly assigns the NCM code for each product transaction based on product description. In order to prove that these models learned useful information from a text description, we compare them to a dummy model that ignores features.

| Model | Base | Accuracy | Accuracy STD | Prediction Time (s) | Prediction Time STD (s) | Training Time (s) | Training Time STD (s) |
|---|---|---|---|---|---|---|---|
| SVM | NF-e | 0.8823 | 0.0 | 18.1544 | 1.5430 | 81.5964 | 3.9396 |
| SVM | NFC-e | 0.8027 | 0.0 | 32.4233 | 0.5984 | 145.6682 | 2.9140 |
| WCNN | NF-e | 0.8604 | 0.0022 | 0.6491 | 0.0833 | 23.0331 | 21.7462 |
| WCNN | NFC-e | 0.7791 | 0.0050 | 0.9744 | 0.2996 | 19.9329 | 2.4455 |
| CCNN | NF-e | 0.8505 | 0.0043 | 1.4879 | 0.4604 | 123.0699 | 10.6410 |
| CCNN | NFC-e | 0.7664 | 0.0116 | 2.5909 | 0.2220 | 442.9317 | 1.86595 |
| Naive Baseline | - | 0,0555 | - | - | - | - | - |

Table 6.1: Summarized Experimental Results for each model and Dataset

From the comparison is clear that the models learned to perform the task, with every model presenting an average accuracy over 0.75, which significantly higher than the naive prediction of 0.05.

Another point of interest was the performance of the term-based model on the short-text classification. The low term count in each document led to the conception that term-frequency representation such as TF-IDF should struggle with short-text problems, such as invoice product descriptions. Models that also learned from the text structure were expected to outperform term count methods. This did not occur in our experiments. While models presented good results for both datasets, the SVM model outperformed both neural network approaches on accuracy,a t the cost of longer training and prediction times.

The argument for a character-based CNN was the possibility that it could positively impact NFC-e classification based on the capacity to better respond to typos and out-of-vocabulary terms. However, either the model failed at addressing these points or they did not matter enough to offset the performance drop due to the increased complexity of the model. On the other side, both CNN models took less time to train and predict than the SVM model.

### 6.1.1 Individual Class results

In this section, individual class results are further explored. Table 6.2 presents the F1-score for randomly sampled models trained on the NF-e dataset, the best results across the three models are presented in bold. The number of samples of each class present in the support column. Performance ranged from 0.68 to 0.99, with at least 10 of the 18

classes presenting scores above 0.9 and only 2 classes presenting scores below 0.75. SVM outperformed both CNN models in most classes.

| Class | SVM F1-SCORE | WCNN F1-SCORE | CCNN F1-SCORE | support |
|---|---|---|---|---|
| 33030010 | 0.8555 | **0.8620** | 0.6127 | 413.0000 |
| 33030020 | **0.7960** | 0.7823 | 0.6673 | 410.0000 |
| 33041000 | **0.9398** | 0.9233 | 0.8936 | 419.0000 |
| 33042010 | **0.9277** | 0.9237 | 0.8656 | 374.0000 |
| 33043000 | **0.9658** | 0.9609 | 0.9221 | 404.0000 |
| 33049100 | **0.9237** | 0.9148 | 0.8967 | 399.0000 |
| 33049910 | **0.6836** | 0.6648 | 0.6002 | 387.0000 |
| 33049990 | **0.7481** | 0.7220 | 0.6416 | 418.0000 |
| 33051000 | **0.9523** | 0.9368 | 0.9325 | 427.0000 |
| 33059000 | **0.8533** | 0.8472 | 0.7594 | 401.0000 |
| 33061000 | **0.9664** | 0.9602 | 0.9603 | 397.0000 |
| 33069000 | **0.9694** | 0.9603 | 0.9366 | 429.0000 |
| 33072010 | **0.8214** | 0.7732 | 0.7455 | 375.0000 |
| 33072090 | **0.7672** | 0.7509 | 0.6503 | 416.0000 |
| 34011190 | **0.9064** | 0.8558 | 0.8245 | 408.0000 |
| 34013000 | **0.8430** | 0.7949 | 0.7544 | 393.0000 |
| 39249000 | 0.9908 | **0.9921** | 0.9778 | 381.0000 |
| 96032100 | 0.9760 | **0.9785** | 0.9595 | 396.0000 |
| accuracy | **0.8823** | 0.8684 | 0.8116 | - |
| macro avg | **0.8826** | 0.8669 | 0.8111 | 7247.0000 |
| weighted avg | **0.8827** | 0.8671 | 0.8109 | 7247.0000 |

Table 6.2: Detailed Class Results for NF-e based on Individual models

Table 6.3 presents F1-Score for randomly sampled models trained on the NFC-e dataset. Performance ranged from 0.52 to 0.94, with 5 classes presenting scores above 0.9 and 5 classes presenting scores below 0.75. Similarly to the NF-e dataset, the SVM model outperformed both CNN models in most classes.

It is possible to notice that the distance in model performance between both datasets varied between classes. This may indicate that the difficulty in classifying NFC-e product descriptions varies with the type of product being described. For certain classes of products, model performance dropped significantly, while for others stayed the same.

## 6.2 Transfer Learning

The goal of this experiment was to validate if it would be possible to utilize a model trained on one type of document to predict the NCM code for the other type. Since there

| Class | SVM F1-SCORE | WCNN F1-SCORE | CCNN F1-SCORE | support |
|---|---|---|---|---|
| 33030010 | **0.6948** | 0.6673 | 0.6108 | 504.0000 |
| 33030020 | **0.7839** | 0.7234 | 0.7492 | 563.0000 |
| 33041000 | **0.8137** | 0.8125 | 0.7875 | 555.0000 |
| 33042010 | **0.7892** | 0.7869 | 0.7410 | 510.0000 |
| 33043000 | **0.9227** | 0.9127 | 0.9064 | 562.0000 |
| 33049100 | **0.7970** | 0.7832 | 0.7638 | 529.0000 |
| 33049910 | 0.5281 | 0.4927 | 0.4875 | 528.0000 |
| 33049990 | **0.6667** | 0.6217 | 0.5725 | 581.0000 |
| 33051000 | **0.8585** | 0.8483 | 0.8406 | 536.0000 |
| 33059000 | **0.7545** | 0.7273 | 0.6955 | 555.0000 |
| 33061000 | **0.9232** | 0.9094 | 0.9093 | 522.0000 |
| 33069000 | **0.9194** | 0.9060 | 0.8873 | 579.0000 |
| 33072010 | **0.7279** | 0.7079 | 0.6600 | 537.0000 |
| 33072090 | **0.6317** | 0.6157 | 0.5720 | 560.0000 |
| 34011190 | 0.9127 | 0.9125 | **0.9186** | 558.0000 |
| 34013000 | **0.8781** | 0.8712 | 0.8668 | 624.0000 |
| 39249000 | 0.8660 | **0.8939** | 0.8522 | 549.0000 |
| 96032100 | **0.9481** | 0.9331 | 0.9396 | 556.0000 |
| accuracy | **0.8027** | 0.7829 | 0.7677 | - |
| macro avg | **0.8009** | 0.7847 | 0.7645 | 9908.0000 |
| weighted avg | **0.8023** | 0.7862 | 0.7662 | 9908.0000 |

Table 6.3: Detailed Class Results for NFC-e based on Individual models

are different availability and processing costs related to each type of document, models that could operate interchangeably could provide a greater degree of freedom to auditors. Also if the model is able to perform well between both types of documents, we have indications that the underlying text representation of that class between these documents is similar.

| Class | SVM F1-SCORE | WCNN F1-SCORE | CCNN F1-SCORE | support |
|---|---|---|---|---|
| 33030010 | **0.5179** | 0.0500 | 0.0720 | 504 |
| 33030020 | **0.6138** | 0.0313 | 0.3439 | 563 |
| 33041000 | **0.7508** | 0.0459 | 0.0603 | 555 |
| 33042010 | **0.6803** | 0.0352 | 0.1256 | 510 |
| 33043000 | **0.8928** | 0.0069 | 0.0355 | 562 |
| 33049100 | **0.7229** | 0.0626 | 0.0679 | 529 |
| 33049910 | **0.3814** | 0.0600 | 0.0734 | 528 |
| 33049990 | **0.4927** | 0.1156 | 0.1181 | 581 |
| 33051000 | **0.8287** | 0.0297 | 0.0120 | 536 |
| 33059000 | **0.6229** | 0.1073 | 0.0673 | 555 |
| 33061000 | **0.8692** | 0.0834 | 0.0273 | 522 |
| 33069000 | **0.8547** | 0.0780 | 0.0187 | 579 |
| 33072010 | **0.6797** | 0.0252 | 0.2122 | 537 |
| 33072090 | **0.5189** | 0.0535 | 0.0273 | 560 |
| 34011190 | **0.8326** | 0.1398 | 0.0035 | 558 |
| 34013000 | **0.7818** | 0.0527 | 0.2085 | 624 |
| 39249000 | **0.2897** | 0.0117 | 0.0000 | 549 |
| 96032100 | **0.9199** | 0.0171 | 0.2113 | 556 |
| accuracy | **0.6837** | 0.0593 | 0.1194 | - |
| macro avg | **0.6806** | 0.0559 | 0.0936 | 9908 |
| weighted avg | **0.6822** | 0.0562 | 0.0947 | 9908 |

Table 6.4: NFC-e dataset Results with models trained on NF-e data.

Table 6.4 presents results in the NFC-e data using models trained on NF-e data. It is important to notice that from all models, only the SVM model managed to preserve its function in this endeavor, with results for both CNN models being similar to random guesses. Regarding the SVM model, there was a considerable drop in scores over all the classes. This can be seen in the accuracy drop of 0.80 to 0.68 over all classes. Nonetheless, this issue was not shared among all classes. Out of the 5 classes with scores above 0.9, only one remained. 2 classes that previously presented scores above 0.75 dropped below 0.4, with one managing to drop from 0.9 to 0.28. 10 out of the 18 classes presented results below 0.75.

This drop in performance may indicate that it is much more difficult to classify retail product descriptions based on the B2B transactions contained in the NF-e data. Classes that dropped harshly in score indicate a mismatch in representation between datasets. It is possible that in one dataset, that class may be represented by a small number of terms shared between training and test sets. The same class is represented by a greater number of terms in the other dataset. This behavior may be explained by the real-world scenario in which a large group of retailers buys from a smaller number of companies. When these retailers resell these products, they describe the same product in new ways. Classes that maintained a good score may indicate that for certain classes the gap between retail and B2B product description is very little.

| Class | SVM F1-SCORE | WCNN F1-SCORE | CCNN F1-SCORE | support |
|---|---|---|---|---|
| 33030010 | **0.5527** | 0.0383 | 0.3203 | 413.0 |
| 33030020 | **0.6082** | 0.0402 | 0.4504 | 410.0 |
| 33041000 | **0.8742** | 0.0246 | 0.0993 | 419.0 |
| 33042010 | **0.8182** | 0.0116 | 0.0827 | 374.0 |
| 33043000 | **0.9389** | 0.0194 | 0.0498 | 404.0 |
| 33049100 | **0.8281** | 0.3219 | 0.1274 | 399.0 |
| 33049910 | **0.4936** | 0.1089 | 0.1020 | 387.0 |
| 33049990 | **0.5788** | 0.1102 | 0.1078 | 418.0 |
| 33051000 | **0.9229** | 0.0416 | 0.0000 | 427.0 |
| 33059000 | **0.7236** | 0.0860 | 0.0729 | 401.0 |
| 33061000 | **0.9300** | 0.0190 | 0.0351 | 397.0 |
| 33069000 | **0.8418** | 0.0982 | 0.0442 | 429.0 |
| 33072010 | **0.6742** | 0.0244 | 0.0000 | 375.0 |
| 33072090 | **0.5441** | 0.1076 | 0.0000 | 416.0 |
| 34011190 | **0.8444** | 0.1927 | 0.0000 | 408.0 |
| 34013000 | **0.7311** | 0.0382 | 0.0301 | 393.0 |
| 39249000 | **0.7581** | 0.0692 | 0.0397 | 381.0 |
| 96032100 | **0.9575** | 0.0041 | 0.1288 | 396.0 |
| accuracy | **0.7538** | 0.0791 | 0.1006 | - |
| macro avg | **0.7567** | 0.0753 | 0.0939 | 7247.0 |
| weighted avg | **0.7570** | 0.0757 | 0.0945 | 7247.0 |

Table 6.5: NF-e dataset Results with models trained on NFC-e data.

Table 6.5 presents results in NF-e data for models trained on NFC-e data. This arrangement also presents a drop in scores throughout all classes. 4 classes maintained a score above 0.9 and only one class presented a drop in score to below 0.5. When comparing results in both datasets, it is possible to see that model this configuration presented better overall scores in all classes than in the previous configuration. No classes

presented drops in the score as sharp as those presented in table 6.4. This may indicate, that NF-e product classification is an easier task than NFC-e product classification and that classifying business transactions using retail data may be easier than the other way around. Retail data presents a larger vocabulary that may contain formal Business terms. Nonetheless, results presented in tables 6.5 and 6.4 provide evidence that, for certain classes, invoice data can be used interchangeably to train NCM classification models.

## 6.3 Comparison between Classification Models

By comparing this models in greater detail, it was possible to see that SVM classifier outperformed their counterparts in nearly all classes. This diverges from the related literature that indicates that term-count based methods should struggle with short-text classification. It was shown that invoice documents can be classified by only a handful of terms and this offsets the challenges of short-text processing.

Transfer Learning results showed that SVM classifier was able to retain the ability of correctly classifying the NCM code for products even when trained with documents of a different type. It was shown that this behavior was common to both datasets, even though there were some loss in performance. Some classes presented a larger drop in performance due to a mismatch in vocabulary between different document uses. The prevalence of SVM may be due to the size of CNN kernels. All kernel sizes were larger than one. This may indicate that while the two types of documents share the same vocabulary, they are worded differently.

Character based models were expected to outperform word-based representation due to its supposed ability to model typos and morphologically similar words. The trade-off would be a increased training time due to a larger network. However, character-based model did not outperform word-based models.

## 6.4 Discussion and Review

This chapter presented experiments results that managed to answer the previously presented questions. It was shown that it is possible to predict the correct NCM code base solely on product description. Models that presented an average accuracy of 88% and 78% on NF-e and NFC-e datasets respectively were successively shown.

# Chapter 7

# Conclusion and Future Work

This work addressed the emerging problem of processing electronic invoice data. E-invoicing is emerging practice with valuable applications and many challenges. In this work, we provided a study on how invoice processing could be tackled. The architecture for SCAN-NF, a invoice classification system to aid tax auditors, was presented and validate through a study case on real world invoice data, in which possible text classification models were tested.

Product-transaction classification was framed as a short-text classification problem, in which the model takes a brief text contained in the product description field of the invoice, and uses this data to assign the NCM code for that product. The NCM code being a standardized code form products and Services in the Mercosur.

In order to guide both studies and development in the field, this work presented a general framework for invoice classification. This framework established a layered structure to invoice processing, in which the output of the previous layer is fed to the next, while also serving as input for valuable applications. Further contributions of this work are contextualized in the presented framework as a product-transaction work. The results of this work could latter be used in both invoice and issuer classification and analysis.

This work present the architecture for SCAN-NF, a system to aid tax auditors by assigning the correct NCM code to product transactions. In order to validate the proposed architecture we outline different research questions: (1) Could the Correct NCM code for a product transaction be predicted solely on product description? (2) How does the available models compare in this task? (3) Can one type of invoice document be used in predicting the other?

In the case study using data provided by the state treasure office it was possible answer these questions. It was confirmed that it was possible to assign NCM code based solely on text description for several classes. Even for the classes that models struggled, accuracy score was superior to the dummy random classifier. Comparisons between models results

showed that while word-based CNN trained and predicted faster than the SVM classifier, the latter performed better on most classes across both datasets. SVM classifiers also managed to preserve knowledge between both types of documents.

The conclusion of this work is that while at product-transaction level invoice classification could be framed as short-text, NCM classification does not share the same challenges as other short-text classification problems. Simple Term-Frequency models outperformed the more complex CNN models on both datasets. This work argues that while product description is brief in invoice documents, the type of product in each invoice can be identified by the presence of a handful of words. Supervised model are valuable resource in this context for being able to leverage the large amount of issuer-labeled documents to create models that are able to identify these words. These models are easier to train and maintain than large rule-base systems.

## 7.1 List of Contributions:

This works contributions may be summarized as :

- Presented a review of invoice processing literature: invoice processing is an emerging subject for many valuable applications, this work presented several related work on how to handle invoice processing in many levels.
- Presented modeling of Invoice Processing as short-text problem: This work presented the characteristics of short-text processing and how classification based on individual product description could be framed as a short-text problem.
- Presented a contextual framework of invoice processing different challenges and opportunities: Since invoice processing is an emerging subject, this work presented and organized, layered framework to guide future research and developments. Main challenges and opportunities for each layer are also presented.
- Presented the architecture for SCAN-NF, a product transaction system to aid tax auditors: this work presented how a NCM classifier could be used by tax-auditors in identifying suspicious transactions and how this model could fit in a larger architecture.
- Presented a study case on real world invoice data on two types of documents: Research questions were validate through experiments made on real world data provided by the state treasury office. These datasets were made public through this work.

- Presented a comparison of different machine learning models over different datasets: different machine learning models were evaluated during experimentation in order to identify different characteristics of these models as well as NCM product Classification.

## 7.2 Future Research:

Future Research could tackle the following points:

- Utilizing mismatches between reported NCM code and predicted NCM code in order to estimate recoverable taxes: While this work provided ways to classify the NCM code, there is still room to evaluate how much could be recovered from a given transaction. This would require a better modelling between the relationship of the NCM code and taxation, the calculation of the range of recoverable taxes and the agent policy for auditing a particular issuer based on the aggregate value of all recoverable taxes over a period o time.
- Aggregate analysis at the invoice level: This work addressed invoice classification at disassociated product transaction level. By leveraging invoice level processing, relationship between different products could be used to track unusual transactions as well as establishing economic agents behavior.
- Create a stack of classification methods to identify ill-intended issuers: while many of the proposed themes address invoice documents at more granular levels, real world application of these resources aim at identifying issuers that have intentionally commit irregularities. Profiling these issuers would probably take an ensemble of multiple classification model to predict different attributes and also graph analysis to study the links between suspicious issuers and other business.
- Model real world transaction operations: While the presented models only took reported invoice data as input, there are several aspects of business operations that could be used to model issuer behavior such as storage and transformation.
- Further explore the classification of particularly important classes: Performance varied between experimental classes. Specialized models could be further trained to handle this classes with the aid of knowledge Bases.
- Utilize product description to reconstruct the pathway of merchandise: As mentioned before, business often transform the merchandise as part of its day-to-day operations. By comparing the input and outputs of transactions of businesses could be used to map outliers.

# Bibliography


[1] Kieckbusch, Diego S., Geraldo P. R. Filho, Vinicius Di Oliveira e Li Weigang: *SCAN-NF: A CNN-based System for the Classification of Electronic Invoices through Short-text Product Description.* Em Mayo, Francisco José Domínguez, Massimo Marchiori e Joaquim Filipe (editores): *Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021, October 26-28, 2021*, páginas 501–508. SCITEPRESS, 2021. `https://doi.org/10.5220/0010715200003058`. xi, 44, 46

[2] He, Y, C Wang, N Li e Z Zeng: *Attention and Memory-Augmented Networks for Dual-View Sequential Learning.* Em *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 125–134, 2020. 1, 3

[3] Chang, W. T., Y. P. Yeh, H. Y. Wu, Y. F. Lin, T S Dinh e I Lian: *An automated alarm system for food safety by using electronic invoices.* PLoS ONE, 15(1), 2020. 1, 3, 42

[4] Zhou, M, X Hu, Y Zhu e P Li: *A novel classification method for short texts with few words.* Em *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, páginas 861–865, 2019. 2, 4, 33

[5] Yue, Y, Y Zhang, X Hu e P Li: *Extremely Short Chinese Text Classification Method Based on Bidirectional Semantic Extension.* Em *Journal of Physics: Conference Series*, volume 1437, 2020. 2, 4, 33

[6] Tang, X, Y Zhu, X Hu e P Li: *An integrated classification model for massive short texts with few words.* Em *ACM International Conference Proceeding Series*, páginas 14–20, 2019. 4, 33

[7] Yu, J, Y Qiao, N Shu, K Sun, S Zhou e J Yang: *Neural Network Based Transaction Classification System for Chinese Transaction Behavior Analysis.* Em *Proceedings - 2019 IEEE International Congress on Big Data, BigData Congress 2019 - Part of the 2019 IEEE World Congress on Services*, páginas 64–71, 2019. 4, 33

[8] Zhu, Y, Y Li, Y Yue, J Qiang e Y Yuan: *A Hybrid Classification Method via Character Embedding in Chinese Short Text with Few Words.* IEEE Access, 8:92120–92128, 2020. 4, 33

[9] Wang, Jin, Zhongyuan Wang, Dawei Zhang e Jun Yan: *Combining knowledge with deep convolutional neural networks for short text classification.* IJCAI International

Joint Conference on Artificial Intelligence, páginas 2915–2921, 2017, ISSN 10450823. 4, 32

[10] Naseem, Usman, Imran Razzak, Katarzyna Musial e Muhammad Imran: *Transformer based Deep Intelligent Contextual Embedding for Twitter sentiment analysis.* Future Generation Computer Systems, 113:58–69, 2020, ISSN 0167739X. `https://doi.org/10.1016/j.future.2020.06.050`. 4, 32

[11] Grida, Mohamed, Hasnaa Soliman e Mohamed Hassan: *Short Text Mining: State of the Art and Research Opportunities.* Journal of Computer Science, 15(10):1450–1460, Oct 2019. `https://thescipub.com/abstract/jcssp.2019.1450.1460`. 4

[12] Kieckbusch., Diego, Geraldo Filho., Vinicius Di Oliveira. e Li Weigang.: *SCAN-NF: A CNN-based System for the Classification of Electronic Invoices through Short-text Product Description.* Em *Proceedings of the 17th International Conference on Web Information Systems and Technologies - WEBIST,*, páginas 501–508. INSTICC, SciTePress, 2021, ISBN 978-989-758-536-4. 5

[13] Aggarwal, Charu C. e Charu C. Aggarwal: *Machine Learning for Text.* 2018. 7, 9

[14] Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu e Pavel Kuksa: *Natural language processing (almost) from scratch.* Journal of Machine Learning Research, 12:2493–2537, 2011, ISSN 15324435. 7

[15] Alsmadi, Issa e Keng Hoon Gan: *Review of short-text classification.* International Journal of Web Information Systems, 15(2):155–182, 2019, ISSN 17440092. 10, 14, 32

[16] Song, Ge, Yunming Ye, Xiaolin Du, Xiaohui Huang e Shifu Bie: *Short Text Classification: A Survey.* Journal of Multimedia, 9(5):635–643, 2014, ISSN 1796-2048. 12

[17] Ellen, Jeffrey: *ALL ABOUT MICROTEXT - A Working Definition and a Survey of Current Microtext Research within Artificial Intelligence and Natural Language Processing.* Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART-2011), páginas 329–336, 2011. 12, 13

[18] Rafeeque, P. C. e S. Sendhilkumar: *A survey on Short text analysis in Web.* 3rd International Conference on Advanced Computing, ICoAC 2011, páginas 365–370, 2011. 13, 14

[19] Ellen, Jeffrey: *Contrasting machine learning approaches for microtext classification.* Proceedings of the 2011 International Conference on Artificial Intelligence, ICAI 2011, 2:543–548, 2011. 13

[20] Abrol, Satyen e Latifur Khan: *TWinner: Understanding news queries with geo-content using Twitter.* Proceedings of the 6th Workshop on Geographic Information Retrieval, GIR'10, páginas 18–19, 2010. 14

[21] O'Connor, Brendan, Michel Krieger e David Ahn: *TweetMotif: Exploratory search and topic summarization for Twitter.* ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media, (May):384–385, 2010. 14

[22] Dela Rosa, Kevin e Jeffrey Ellen: *Text classification methodologies applied to micro-text in military chat.* 8th International Conference on Machine Learning and Applications, ICMLA 2009, páginas 710–714, 2009. 14

[23] Sahami, Mehran e Timothy D. Heilman: *A web-based kernel function for measuring the similarity of short text snippets.* Proceedings of the 15th International Conference on World Wide Web, páginas 377–386, 2006. 14, 31

[24] Yih, Wen Tau e Christopher Meek: *Improving similarity measures for short segments of text.* Proceedings of the National Conference on Artificial Intelligence, 2:1489–1494, 2007. 14, 31

[25] Sriram, Bharath, David Fuhry, Engin Demir, Hakan Ferhatosmanoglu e Murat Demirbas: *Short text classification in twitter to improve information filtering.* SIGIR 2010 Proceedings - 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (January 2010):841–842, 2010. 14

[26] Qiang, Jipeng, Zhenyu Qian, Yun Li, Yunhao Yuan e Xindong Wu: *Short Text Topic Modeling Techniques, Applications, and Performance: A Survey.* IEEE Transactions on Knowledge and Data Engineering, 14(8):1–1, 2020, ISSN 1041-4347. 14

[27] Goodfellow, Ian, Yoshua Bengio e Aaron Courville. 2016. 15, 16, 18

[28] Lecun, Yann, Yoshua Bengio e Geoffrey Hinton: *Deep learning.* Nature, 521(7553):436–444, 2015, ISSN 14764687. 15, 16, 17

[29] Young, Tom, Devamanyu Hazarika, Soujanya Poria e Erik Cambria: *Recent trends in deep learning based natural language processing [Review Article].* IEEE Computational Intelligence Magazine, 13(3):55–75, 2018, ISSN 15566048. 15, 20, 21

[30] Widrow, Bernard e Michael A. Lehr: *30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation.* Proceedings of the IEEE, 78(9):1415–1442, 1990, ISSN 15582256. 16

[31] Kingma, Diederik P e Jimmy Lei Ba: *ADAM.* Iclr, páginas 1–15, 2015. `https://arxiv.org/pdf/1412.6980.pdf{%}22entiredocument`. 17

[32] Duchi, John, Elad Hazan e Yoram Singer: *Adaptive subgradient methods for online learning and stochastic optimization.* COLT 2010 - The 23rd Conference on Learning Theory, 12:257–269, 2010. 17

[33] Kim, Yoon: *Convolutional neural networks for sentence classification.* EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, (2011):1746–1751, 2014. 19, 20, 32

[34] Elman, Jeffrey L.: *Finding structure in time.* Cognitive Science, 14(2):179–211, 1990, ISSN 03640213. 20

[35] Hochreiter, Sepp e Jürgen Schmidhuber: *Long Short-Term Memory.* Neural Computation, 9(8):1735–1780, 1997, ISSN 08997667. 20

[36] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk e Yoshua Bengio: *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, páginas 1724–1734, 2014. 21

[37] Harris, Zellig S.: *Distributional Structure.* WORD, 10(2-3):146–162, 1954, ISSN 0043-7956. 21

[38] Wang, Xin, Yuanchao Liu, Chengjie Sun, Baoxun Wang e Xiaolong Wang: *Predicting polarities of tweets by composing word embeddings with long short-Term memory.* ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference, 1:1343–1353, 2015. 21

[39] Graves, Alex, Abdel Rahman Mohamed e Geoffrey Hinton: *Speech recognition with deep recurrent neural networks.* ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, (3):6645–6649, 2013, ISSN 15206149. 21, 22

[40] Bahdanau, Dzmitry, Kyung Hyun Cho e Yoshua Bengio: *Neural machine translation by jointly learning to align and translate.* 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, páginas 1–15, 2015. 22, 23

[41] Rumelhart, D. E., G. E. Hinton e R. J. Williams: *Learning Representations by Error Propagating Errors.* Nature, (323):533–536, 1986. 24

[42] Bengio, Yoshua, Réjean Ducharme e Pascal Vincent: *A neural probabilistic language model.* Advances in Neural Information Processing Systems, 3:1137–1155, 2001, ISSN 10495258. 24

[43] Mikolov, Tomas, Kai Chen, Greg Corrado e Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space.* páginas 1–12, 2013. `http://arxiv.org/abs/1301.3781`. 24, 25

[44] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado e Jeffrey Dean: *Distributed representations ofwords and phrases and their compositionality.* Advances in Neural Information Processing Systems, (October 2013), 2013, ISSN 10495258. 24

[45] Mikolov, Tomáš, Jiří Kopecký, Lukáš Burget, Ondřej Glembek e Jan Honza Černocký: *Neural network based language models for highly inflective languages.* ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, páginas 4725–4728, 2009, ISSN 15206149. 26

[46] Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch e Armand Joulin: *Advances in pre-training distributed word representations*. LREC 2018 - 11th International Conference on Language Resources and Evaluation, (1):52–55, 2019. 25

[47] Bojanowski, Piotr, Edouard Grave, Armand Joulin e Tomas Mikolov: *Enriching Word Vectors with Subword Information*. Transactions of the Association for Computational Linguistics, 5:135–146, 2017, ISSN 2307-387X. 25

[48] Pennington, Jeffrey, Richard Socher e Christopher D. Manning: *GloVe: Global Vectors for Word Representation*. Em *Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1532–1543, 2014. `http://www.aclweb.org/anthology/D14-1162`. 26

[49] Chiu, Billy, Anna Korhonen e Sampo Pyysalo: *Intrinsic Evaluation of Word Vectors Fails to Predict Extrinsic Performance*. páginas 1–6, 2016. 27

[50] Faruqui, Manaal, Yulia Tsvetkov, Pushpendre Rastogi e Chris Dyer: *Problems With Evaluation of Word Embeddings Using Word Similarity Tasks*. páginas 30–35, 2016. 27, 35

[51] Conneau, Alexis e Douwe Kiela: *SentEval: An evaluation toolkit for universal sentence representations*. LREC 2018 - 11th International Conference on Language Resources and Evaluation, páginas 1699–1704, 2019. 28

[52] Yin, Wenpeng e Hinrich Schütze: *Learning Meta-Embeddings by Using Ensembles of Embedding Sets*. (2013), 2015. `http://arxiv.org/abs/1508.04257`. 28

[53] Coates, Joshua N. e Danushka Bollegala: *Frustratingly easy meta-embedding-computing meta-embeddings by averaging source word embeddings*. NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 2:194–198, 2018. 28

[54] Kiela, Douwe, Changhan Wang e Kyunghyun Cho: *Dynamic Meta-Embeddings for Improved Sentence Representations*. `https://arxiv.org/abs/1804.07983`. 28

[55] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser e Illia Polosukhin: *Attention is all you need*. Advances in Neural Information Processing Systems, 2017-Decem(Nips):5999–6009, 2017, ISSN 10495258. 29, 30

[56] Phan, Xuan Hieu, Le Minh Nguyen e Susumu Horiguchi: *Learning to classify short and sparse text & web with hidden topics from large-scale data collections*. Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08, (January):91–99, 2008. 31, 34

[57] Alsmadi, I e G K Hoon: *Term weighting scheme for short-text classification: Twitter corpuses*. Neural Computing and Applications, 31(8):3819–3831, 2019. 32

[58] Wang, Sida e Christopher D. Manning: *Baselines and bigrams: Simple, good sentiment and topic classification*. 50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference, 2(July):90–94, 2012. 32

[59] Zhang, Xiang e Yann LeCun: *Text Understanding from Scratch*, 2016. http://arxiv.org/abs/1502.01710. 32

[60] Chen, J, S Yan e K. C. Wong: *Verbal aggression detection on Twitter comments: convolutional neural network for short-text sentiment analysis*. Neural Computing and Applications, 32(15):10809–10818, 2020. 33

[61] Paalman, J, S Mullick, K Zervanou e Y Zhang: *Term based semantic clusters for very short text classification*. Em *International Conference Recent Advances in Natural Language Processing, RANLP*, volume 2019-Septe, páginas 878–887, 2019. 34, 42

[62] ENCAT: *Manual de Orientação do Contribuinte - Padrões Técnicos de Comunicação*. 38, 39

[63] Feng, Yanhui, Peng Jiang, Zhenyu Gu e Yonghui Dai: *Study of recognition of electronic invoice image*. IEEE Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2021, páginas 1582–1586, 2021. 42

[64] Zhang, H, B Dong, B Feng, F Yang e B Xu: *Classification of Financial Tickets Using Weakly Supervised Fine-Grained Networks*. IEEE Access, 8:129469–129477, 2020. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85089215581{&}doi=10.1109{%}2FACCESS.2020.3007528{&}partnerID=40{&}md5=9fffb4e8a98ac64be2fa28de21f4e632. 42

[65] Palm, R B, F Laws e O Winther: *Attend, copy, parse end-to-end information extraction from documents*. Em *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, páginas 329–336, 2019. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85079851980{&}doi=10.1109{%}2FICDAR.2019.00060{&}partnerID=40{&}md5=29b092a6c8a3c0caf86779867d63d202. 42

[66] Tang, Peng, Weidong Qiu, Zheng Huang, Shuang Chen, Min Yan, Huijuan Lian e Zhe Li: *Anomaly detection in electronic invoice systems based on machine learning*. Information Sciences, 535:172–186, 2020, ISSN 00200255. https://doi.org/10.1016/j.ins.2020.03.089. 42

[67] Marinho, Mayara C., Vinicius Di Oliveira, Sérgio A. P. B. Neto, Li Weigang e Vinícius R. P. Borges: *Visual Analysis of Electronic Invoices to Identify Suspicious Cases of Tax Frauds*. Em Rocha, Álvaro, Carlos Ferrás, Abel Méndez Porras e Efren Jimenez Delgado (editores): *Information Technology and Systems*, páginas 185–195, Cham, 2022. Springer International Publishing, ISBN 978-3-030-96293-7. 42

[68] Schulte, Johannes P., Felipe T. Giuntini, Renato A. Nobre, Khalil C. do Nascimento, Rodolfo I. Meneguette, Weigang Li, Vinícius P. Gonçalves e Geraldo P. Rocha Filho: *ELINAC: Autoencoder Approach for Electronic Invoices Data Clustering*. Applied

Sciences, 12(6), 2022, ISSN 2076-3417. https://www.mdpi.com/2076-3417/12/6/3008. 42

[69] Agapito, Giuseppe, Barbara Calabrese, Pietro Hiram Guzzi, Sabrina Graziano e Mario Cannataro: *Association Rule Mining from large datasets of clinical invoices document*. Proceedings - 2019 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2019, páginas 2232–2238, 2019. 42

[70] Da Rocha, Cristiano Cortez, Márcio Parise Boufleur, Leandro Da Silva Fornasier, Júlio César Narciso, Andrea Schwertner Charão, Vinícius Maran, João Carlos D. Lima e Benhur O. Stein: *SQL query performance on hadoop: An analysis focused on large databases of brazilian electronic invoices*. ICEIS 2018 - Proceedings of the 20th International Conference on Enterprise Information Systems, 1(Iceis):29–37, 2018. 43

[71] Cuylen, Angelica, Lubov Kosch e Michael H. Breitner: *Development of a maturity model for electronic invoice processes*. Electronic Markets, 26(2):115–127, 2016, ISSN 14228890. 43