Master thesis
Computer Science

Radboud University

Classifying Short Text for the Harmonized System with Convolutional Neural Networks

Author:
Jeffrey Luppes
s4699424
jeffluppes@gmail.com

First supervisor/assessor:
Prof.dr.ir. Arjen P. de Vries
arjen@cs.ru.nl

Second assessor:
Faegheh Hasibi
f.hasibi@cs.ru.nl

August 23, 2019

Abstract

Classifying goods in the hierarchical Harmonized System is a challenging problem for both humans and machines. To combat this, we propose a Convolutional Neural Network (CNN) architecture to label descriptions on the basis of short text descriptions. In order to obtain domain-specific word embeddings we identify and collect text data from several online resources, including DBpedia. We evaluate our approach on two data data sets, and find we can classify the HS-2 code (chapter) with an f1-score of 0.92 and 0.90, respectively. On the much more complex HS-4 (heading) level we find we can still classify among over 1200 classes with an accuracy of 0.88 and 0.84. We believe our gains are partly due to our architecture, which includes convolutional layers with filters of 2 to 5 words, batch-normalization, and more fully-connected layers than previous work. We theorize that this approach can also be applied to classify sub-headings and beyond and provides a strong baseline in this domain.

Acknowledgements

This master thesis project owes a great deal to Atos, as without their resources and data, this project would not have been possible.

I would like to thank a number of people at Atos in particular. First and foremost, a thank-you goes out to Jorrit de Vries for his support, sanity checks, endless VM (re-)booting, and troubleshooting with me. Jorrit, your patience and your ability to go above and beyond, often outside normal office hours and during holiday breaks meant I could squeeze in a lot of extra hours of model training. Without you, I would not have been able to move forward with my work. Additionally, I would like to thank Yaroslav Logachev of Atos France for his domain expertise and insights. A special thanks also to Arjen Tuinstra, for matching me with this project and giving me the opportunity to work at a high level, and even allowing me to present my work in front of a large group of the expert community at Atos.

Lastly, I'd like to thank Arjen de Vries, my supervisor, for without his guidance and inspiration I would not have been able to complete this thesis. Arjen, you've been one of the most approachable and friendly professors I've met, and I hope you continue to share that kindness with your students.

Contents

Chapter 1

Introduction

1.1 General Introduction

The world increasingly relies on logistic processes to supply and provide for a ever-growing population. Facilitating global trade is a shared customs standard which classifies goods with a numerical code, allowing every customs authority to implement the same nomenclature. This standardized system is the Harmonized System (HS) which classifies goods in a hierarchical code framework. These codes appear on standard freight documents such as waybills and Single Administrative Documents (SAD). At the core of these documents is a provision for a textual description of goods. An example of a description alongside a valid HS code is shown in 1.1. The burden of providing the correct code and a valid text description is placed on the trader, who has to supply and ascertain that both of these are correct. However, due to the complexity of the system this classification process remains hard to do for non-experts, with some agencies estimating a rate of miss-classification of between 17% and 30% [17]. More detailed information on the Harmonized system itself and the consequences of miss-classification can be found in the last section of this chapter.

Description	HS Code
Parts of filtering or purifying machinery	842199

Table 1.1: An example of a (corrected) HS-classification. The HS code shown here is defined at the HS-6 level, which defines over 5000 possible labels.

Classification on the basis of these texts to find the correct HS label is a complex problem, due to a number of problems:

- The textual descriptions are brief, creating a sparse feature space for

classification

- These descriptions do not follow natural language syntax and often lack the structure otherwise found in text
- The number of possible labels for a text description is rather large compared to other text classification problems, at over 5000 at the highest level of detail
- Keywords such as "diesel fuel" that may be used to identify a product appear both when defining a raw form of the item, as well as parts of products such as waxes or engines ("diesel engine"), or even when describing complete vehicles
- Not all product labels are as frequent as others, making it hard to gather data on edge cases
- Misspellings and grammatical errors are common [34] and are hard to correct without more context
- The text descriptions often contain special domain-specific words specific to shipping, such as the abbreviations *stc*, "said to contain", and *fcl*, which stands for "full-container load"
- The occurrence of *Hapax legomena*, words or phrases, or even complete descriptions, that only appear once, is frequent.

We take on a fairly applied angle in this master thesis. In order to attempt to provide a computational solution to this problem we focus on the use of 1-dimensional convolutional neural networks (CNNs), a deep learning technique for text classification introduced by Kim [27], which gained traction as a competitive method for text classification able to expand to large amounts of data and a sprawling label space.

Our contributions are two-fold. We find that we can classify products descriptions with CNNs with good success, and we obtain domain-specific word embeddings by iterative pre-training on context-rich, domain-specific corpora we obtain from DBpedia, scraping Wikipedia and public data sets. The corpora we create contain almost two billion tokens. Our approach is largely adapted from existing architectures and includes the use of batch normalization as introduced by Ioffe and Szegedy [21] instead of dropout and pooling techniques. Our approach achieves significant gains over conventional methods such as linear SVMs, binary multinomial naive Bayes classifiers, and logistic regression, but also Kim's original work when applied to our problem. We hypothesize our product description architecture can also be applied to other short text problems.

1.2 Research focus

In this work, we aim to classify goods descriptions on the HS-2 and HS-4 level. While the HS-2 level is arguably high-level and has therefore limited use for customs purposes, the HS-4 level sees significant use in summary declarations.

Our main research question is formulated as follows: **can we define a CNN architecture that can classify HS codes based on short descriptions?** For this we will also establish a number of baseline methods to compare and evaluate performance.
When operating on the word-level, CNN architectures frequently use word embeddings to be able to process words as high-dimensional features. While the overall quality of embeddings increases with the amount of available data, finding domain-specific corpora eliminates the ambiguity encountered when using general embeddings. Further motivation for this is found in the data we encounter in this domain. For instance, a *slasher* is exclusively a weed-cutting machete-like tool within our domain, and not the film genre. Furthermore, *caterpillar* is more likely to refer to a brand than the larvae of many insect species. We therefore focus on training embeddings on domain-specific sources outside of our original data source. This leads to our sub-question: **What is the added benefit of context-rich domain-specific pre-training of word embeddings for a convolutional neural network to classify short product descriptions?** To this end we identify context-rich domain specific text data on the web, and analyze what we gain by using this data for learning word embeddings.

1.3 Background: The Harmonized System

The Harmonized System (HS) is a world-wide standard for product classification and sees an almost complete adaption worldwide [7]. It is a complex hierarchical system, with multiple levels that divide goods into chapters, headings, and finally subheadings. While the top-most level only contains up to 99 definitions, the system allows for over 5300 precise definitions at level of subheadings. Furthermore, countries have the full liberty to extend upon subheading definition with one or two more levels for their own internal needs [56]. Examples of these extensions are the Brazilian *NCM* and the United States' *HTS*. It is important to note that at every level the system is defined by both a code and a textual description, and that at each level the definitions become more detailed.

1.3.1 Examples

In the Harmonized System, a sand-moving diesel-fuel using bulldozer falls under the chapter of *nuclear reactors, boilers, machinery and mechanical appliances, or parts thereof.* This chapter has the code 84. One level lower is the description of *self-propelled bulldozers, angledozers, graders, levellers, scrapers, mechanical shovels, excavators, shovel loaders, tamping machines, and road rollers.* The harmonized system code description for this heading is 8429. The system then still goes deeper when defining the subheading group of *self-propelled bulldozers and angledozers other than track-laying.* This description matches the code 842919. Countries are allowed to extend this system with their own codes, which are added as a suffix to the subheading code (HS-6). This is illustrated in table 1.2.

Level	Example Code
Chapter (HS-2)	84
Heading (HS-4)	8429
Subheading (HS-6)	842919
Country-specific extension (HS-8 and beyond)	84291900

Table 1.2: The hierarchical nature of the Harmonized System demonstrated

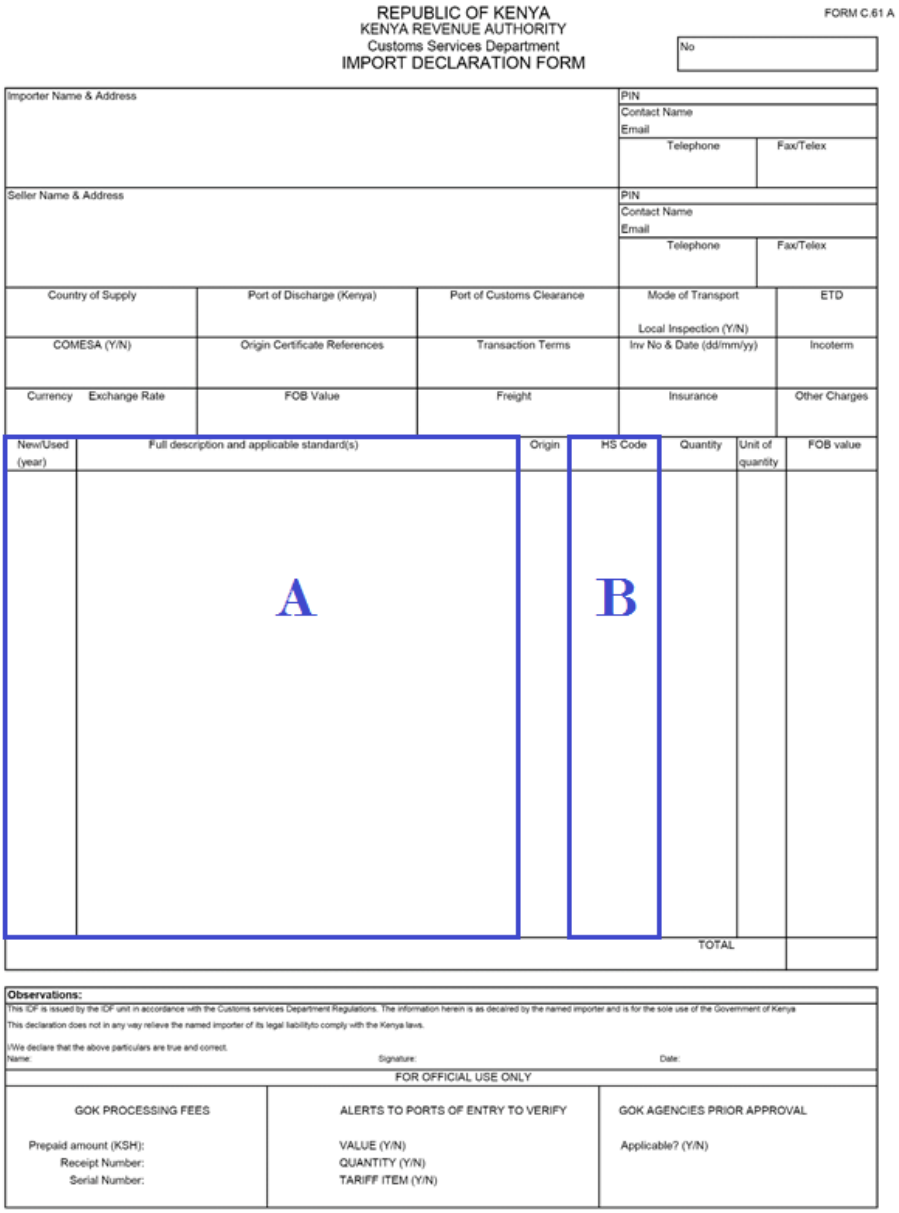
For the sake of consistency, we will refer to these descriptions according to the length of their corresponding HS-code. Thus, the chapter will be referred to as HS-2, the heading as HS-4, and the subheading as HS-6. In the case of country-specific data with a code consisting out of 8 digits, we will refer to these as HS-8. While the classification system includes 21 sections as well as a super-group of chapters, this is outside of the scope of this research as these are not tied to the Harmonized System's code itself and have limited relevancy for classification.

1.3.2 Usage of the Harmonized System

Upon entering a country, carriers are required to present a customs form for their cargo. An example of such a form is shown in 1.1, with the fields for product descriptions marked. Other countries or custom unions have their own forms, but the general usage is, due to the high degree of adaption of the harmonized system, essentially the same. This form contains information on the shipment, and is often the first item considered when evaluating which shipments should be inspected if it so happens that doubts are raised regarding the truthfulness of the declaration. The form specifically has fields for a text describing the cargo and its corresponding classification, written by traders or agents.

Another example of a HS-code as it is likely to appear in its use for the consumer market can be seen in Figure 1.2. This sticker includes an European Combined Nomeclature (CN) subheading code of eight digits (HS-8) along with a product description. Interestingly, the code consists out of eight digits. This means that we can infer that the product was imported from inside the EU, as imports from outside the EU are given a TARIC code (corresponding to HS-10) instead.

Lastly, HS classification codes serve as a way to collect statistics on what is imported and exported to a country. Although this is a significant focus of the Harmonized System, it is outside of the scope of this thesis.


REPUBLIC OF KENYA
KENYA REVENUE AUTHORITY
Customs Services Department
IMPORT DECLARATION FORM

FORM C.61 A

No

Importer Name & Address | PIN | Contact Name | Email | Telephone | Fax/Telex

Seller Name & Address | PIN | Contact Name | Email | Telephone | Fax/Telex

Country of Supply | Port of Discharge (Kenya) | Port of Customs Clearance | Mode of Transport | Local Inspection (Y/N) | ETD

COMESA (Y/N) | Origin Certificate References | Transaction Terms | Inv No & Date (dd/mm/yy) | Incoterm

Currency | Exchange Rate | FOB Value | Freight | Insurance | Other Charges

New/Used (year)	Full description and applicable standard(s)	Origin	HS Code	Quantity	Unit of quantity	FOB value
	A		B			
					TOTAL	

A

B

Observations:

This IDF is issued by the IDF unit in accordance with the Customs services Department Regulations. The information herein is as declared by the named importer and is for the sole use of the Government of Kenya

This declaration does not in any way relieve the named importer of its legal liabilityto comply with the Kenya laws.

I/We declare that the above particulars are true and correct.

Name: Signature: Date:

FOR OFFICIAL USE ONLY

GOK PROCESSING FEES	ALERTS TO PORTS OF ENTRY TO VERIFY	GOK AGENCIES PRIOR APPROVAL
Prepaid amount (KSH): Receipt Number: Serial Number:	VALUE (Y/N) QUANTITY (Y/N) TARIFF ITEM (Y/N)	Applicable? (Y/N)

Figure 1.1: The Kenyan customs form 61a, which is used to declare imports. Outlined are two relevant sections to describe products and assign a HS-code. Forms like these are used world-wide. Section A provides a text field for a product description, and section B provides a field for the HS code. Other fields on the form are not used within this thesis, but one could argue they can easily be utilized to improve performance on classification tasks.

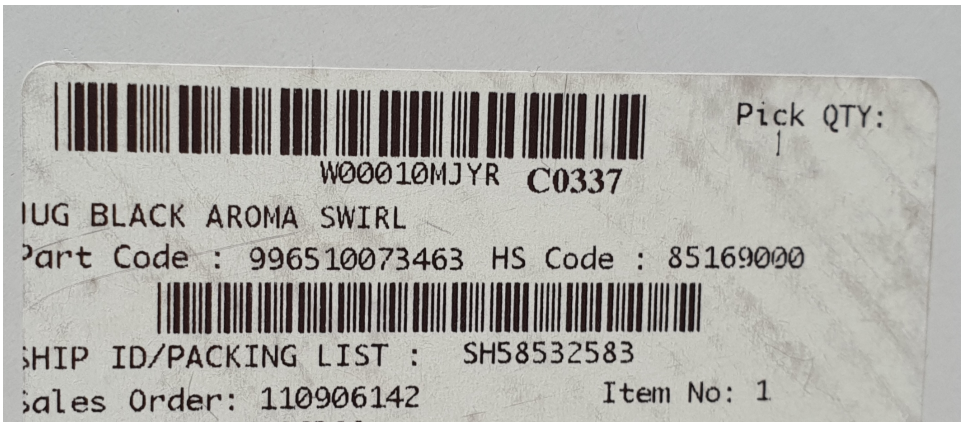


Figure 1.2: A HS code as it is likely to appear with documents accompanying packages. This sticker contains a textual description ('JUG BLACK AROMA SWIRL') and a HS-8 code (85169000) which corresponds to parts of *electric instantaneous or storage water heaters and immersion heaters; electric space-heating apparatus and soil-heating apparatus; electrothermic hairdressing apparatus (for example, hairdryers, hair curlers, curling tong heaters) and hand dryers; electric smoothing irons; other electrothermic appliances of a kind used for domestic purposes; electric heating resistors, other than those of heading 8545.* The item imported was a coffee pot.

Chapter 2

Related Work

Text Classification and Word Representations

Text Classification is a common supervised learning task in which a body of text, or document containing text, needs to be assigned to a set of class labels. Common applications of text categorization include article categorization, spam detection, sentiment analysis, and authorship attribution [50, 11, 51]. Pivotal work in this area includes Joachim's approach that utilized Support Vector Machines (SVMs), which has remained influential [23]. Apart from SVMs, multinomial- and Bernouilli-kernelled Naive Bayes (NB), and Logistic Regression are often utilized for these problems [11, 55]. However, in order to enable algorithms to take advantage of the information represented by words, we first need to create a representation of text that algorithms can work with. Common feature-generating methods such as the Bag-of-Words (BoW) model, ngrams, the term frequency-inverse document frequency (tf-idf), and word vectors have been used with great effect [40].

In the BoW model, originally introduced by Harris [18] every word gets represented by an index in a vector, which can thus grow incredibly spare with the vocabulary used. A sentence of natural language is then simply represented by a sparse vector containing counts of each word appearing in the vocabulary. A problem is that this vocabulary grows larger the more unique words appear in a text.

On a conceptual level, ngrams are similar to the BoW approach and can be either applied to words or individual characters. The idea is, to look at adjacent words and take counts of words appearing next to each other, instead of focusing on singular words. For instance, a n-gram approach involving bigrams (2-grams) takes two words together, and produces a feature vector that represents a sentence by these counts. A sliding window approach is used to move across a sentence. Both BoW and ngrams are relatively naive

methods of dealing with text, as all words or ngrams are considered to be equally important.

The tf-idf weighting method tries to address this problem by multiplying a how often a term appears in a document with a log function of how many documents there are in the collection divided by how many document contain the term. If a word is common across all documents this assigns a lower weight than words that are rare across all documents, meaning that rarer words which are theorized to contribute more to the meaning of a document contribute more. The formula for tf-idf is shown below, where tf is the term frequency of term t in document d, N is the total number of documents in the corpus, and df is the document frequency of term t.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \log \frac{N}{\text{df}_t}$$

This has a shortfall in that we still treat words as simply random-ordered features present in a document. Indeed, speakers of natural language will know that word order is actually very important and yet this information is simply lost when converting text to an tf-idf feature vector because it loses the order in which words appear.

A different take on word vector representations comes from word embeddings, popularized by methods such as word2vec [41]. Word embeddings are a representation of a word as a single vector, and the idea has in fact been around for longer [4]. These vector space methods often involve a shallow neural network to produce word embeddings learned from processing large amounts of data [41, 32, 16]. Commonly, these are 50 to 300-dimensional vectors, although more recent work incorporates much higher dimensionality [13, 46]. As Word2vec was one of the first such methods to become popular, it has been well-studied as a result. The appeal and relative simplicity means it can readily be used for classification, recommendation, and similarity matching [26]. Word embedding methods quickly became a state-of-the-art staple in text classification [46, 20]. However, for classical methods such as SVMs, it appears that using word embeddings decreases performance [35].

A lot of work has been done to evaluate word embeddings as well to develop new embedding approaches. There many adaptions and similar methods available such as doc2vec [31], syntax-aware wang2vec [37], fastText [5], and Stanford's log bi-linear Glove, which does not rely on a shallow neural network [45].
A key idea is that these embeddings can be learned on data outside the target data set. This is an elegant way to take information from a corpus such as, for instance, a Wikipedia dump and use it to collect contextual

information on word occurrences, which can then be applied to, for instance, a sentiment analysis task based on the short text gathered from internet comments. However, the word embeddings learned from data will carry any implicit biases present in their training texts [6], although one could argue this is exactly the type of similarities we intend to capture in the harmonized system text domain. It remains hard to apply word embeddings to scenarios where training data is limited or consisting of short messages. In this line, work has been done to increase performance with data from another domain [1].

Short Text Classification

A specific sub-field of text categorization deals with short text. Short text is especially prevalent online, where it can take the form of twitter messages, reviews, titles, chat messages, URLs, and search queries. Due to the limited information contained in a message the problem is characterized by data sparsity and a lack of context [54]. This is especially true with regards to Twitter messages (also known as Tweets) and chat messages, which are often limited in the amount of characters or words in a sentence. Recurring problems are also that these texts do not usually follow natural language conventions: in order to stay brief, these texts often contain a telegram-style syntax and common natural language processing techniques do not apply in the same way they do with spoken text, or longer text as is common in books. This results in a problem of ambiguity, where a piece of text might have multiple meanings but does not provide enough context to figure out which meaning was intended. Often, these texts also include specific tokens and characters unique to the domain. Interestingly, it has been shown that Naive Bayes models outperform SVMs if the text length is limited, while for longer text the opposite still holds true [55].

Product Classification

With regards to product classification, we observe many of the same problems as in natural short text. A text description for a product is often of similar telegram-style [58], often just limited to a product title. Furthermore, it may include odd tokens such as shop keeping units, brand names, and physical dimensions of the package or pallet it is packed with. It suffers from the same problem of sparsity as encountered with short text. This domain has a significantly overlap in methods with recommendation, where the goal is matching the information need of an user with the best matching product from an inventory.

Often, product classification involves a large amount of classes. Many systems used in e-commerce are inherently multi-label with classes that are not mutually exclusive and are part of some taxonomy [28]. Convolutional Neural networks have already been applied in this area due to their success with the extreme inherent class label space [38].

The Harmonized System

There have only been a couple of papers that discuss the Harmonized System and HS code prediction problem. In spite of that, we have strong suspicions that the amount of unpublished work done on this problem is actually far greater, due to the business need for classification for global trade and due to patented solutions in this space. We also consider an overview of patents to be outside of the scope of our work, after failing to identify any patented solution that approached our work.

In 2017, Batista published a paper that focused on NCM code prediction within two specific chapters: 20 and 90 [12]. The Brazilian NCM system extends the six digits of the harmonized system. This approach utilized a naive Bayes based on tf-idf weighted features. They were able to obtain a Portuguese-language data sets from the Brazilian Ministry of Economy, and evaluated their approach against it with good success. They found that within the chapters of 20 (59 categories) and 90 (819 categories) they could predict labels within these classes with 96 and 88 percent accuracy respectively.

This focus on specific chapters is also seen in Ding et al's [14] earlier 2015 work, who used a vector space approach. Their data set was obtained from the Singaporean Customs office. Regardless of their great performance within a chapter, one of their main findings was that the shortness of the descriptions available to them made it hard to classify within a chapter: when classifying a text of three tokens or less, they could only obtain 15.86% accuracy.

A recent paper by Li and Li appears to be the first to approach the problem by means of a neural network [33]. In their work they construct two simple CNNs, one for text and one for images, to classify descriptions of shoes into four classes based on six HS-8 code categories. They do not attempt to directly classify HS codes. Their data set consisted of 10,000 images of shoes alongside textual descriptions as well as the height of the shoe shaft, which were scraped from a Chinese trade website.

To the best of our knowledge, no author has published work focusing on all

HS-2 or HS-4 chapters at once. Existing work is narrow and often focuses on a single chapter, or two. We believe this may be due to the large amount of classes involved in classification, and the diverse nature of the token space which greatly limits what approaches can be used.

Neural Networks for Text

Neural networks for natural language processing have experienced a surge in popularity in recent years. The most common methods can be roughly grouped in two categories: Convolutional Neural Networks (which includes the approach in this thesis) and Recurrent Neural Networks (which include Long-Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs)).

Recurrent neural networks are not an odd choice, as they excel with modelling sequential data. They have been utilized for language model problems and have in general been applied with success [42, 8, 29, 20, 57]. However, they were initially not well-suited to deal with correlations that are further apart in sentences, due to the vanishing gradient problem during back-propagation where the gradient propagated through the network becomes close to zero when more time steps are made.

With word embeddings, neural networks for text gained a significant boost in popularity and researchers such as Kim [27] and Kalchbrenner[25] soon released their work with feed-forward one-dimensional convolutional neural networks based on word embeddings. The general architecture is much simpler than the convolutional neural networks used in computer vision, as the architectures used often consist of just an embedding layer, a convolutional layer, a max-pooling layer and a varying amount of dense layers for each of a set of pre-defined filter sizes. Interestingly, this shallow and wide architecture remains a impressive baseline [30]. The idea behind the convolutional layers' is that this is akin to taking ngrams with varying convolutional filter sizes [60, 22]. Pooling strategies have been shown to help, as they eliminate poorer ngrams and bring down the training complexity of the network; however, others opt out of pooling completely [27, 60]. Some architectures use multiple filter sizes in a multi-channel approach [27, 38, 16].

Current approaches also include bi-directional LSTMs, which pass over text both from left-to-right and right-to-left. This greatly enhances the ability of RNNs to deal with longer dependencies in text. Among these, the ELMo model [46] successfully applies this concept to produce embeddings that change on differing sentences (context).

Attention mechanisms are also popular and are currently popular choices to capture dependencies in text [36, 47]. One specific attention mechanism is the transformer, an encoder-decoder architecture that involves an attention mechanism to capture dependencies in text [53]. BERT [13] successfully combines this idea with a Masked Language Model objective to produce a complex language model that can be applied to many downstream tasks. It is worth noting that the concept of pre-training on outside data and applying a trained model to supervised tasks is still adhered to with these models. However, with these developments it seems that the concept of transfer learning in language models is definitely shifting from mere word embeddings to more sophisticated models that have been trained for a long period of time, which is very much like the trend in computer vision.

Lastly, it is possible to forsake word-level embeddings completely and construct a convolutional neural model based on characters [59, 10, 51]. These models rival the performance of word-embedding CNNs, but despite their elegance require a deeper network and a larger training time budget [30, 10, 2].

Chapter 3

Methods

This section details the models developed and the methodology we followed. As this is a thesis, we opt to include a more comprehensive description of our Convolutional Neural Network approach in section 3.3.1. Readers well-familiar with these may skip large parts of this section.

3.1 Evaluation Strategy and Metrics

In order to evaluate our approach we use a train/test split of 0.8/0.2 in instances where it is not possible or desirable due to computation time to report cross-validation scores. Where applicable we further sample validation data from the training set, of which a description is included in the sub-section dedicated to each algorithm, below. As the data set suffers from imbalance, for the HS-2 level we report success using the recall, precision, and the (macro weighted) F1-score based on the true positives (*tp*), true negatives (*tn*), false positives (*fp*), and false negatives (*fn*):

$$Recall = \frac{tp}{tp+fn} \qquad Precision = \frac{tp}{tp+fp} \qquad F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

For HS-4 we report the results using the test-time accuracy as a metric as defined in [44]. The motivation is that we encounter the problem of the f1-score being ill-defined when there are no predicted samples for a class, which can occur frequently with the number of labels and potential for the absence of training or evaluation data for a class. The formula for the accuracy is given below:

$$Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$$

3.2 Baseline Methods

3.2.1 Support Vector Machines

Support Vector Machines (SVMs) were introduced as a method for high-feature text classification by Joachims in 1998 [23]. Similar to Joachims' approach, which was limited by 10000 features, we limit the amount of features to 20000. Arguably, the feature space could be much larger, but we are also constrained by the number of class labels. We employ a one-versus-all approach, and thus are actually training n classifiers for n classes. Because this method scales poorly to the number of data points available, we randomly sample 100,000 data points from the data set for training. Similarly, we are also required to reduce the number of data points to 100,000 in the test data in order to obtain predictions on the validation data. This is a considerable concern for the validity and the applicability of our approach, as results may not translate well to performance on all of the data.
We implement both a naive normalized Bag-of-Words SVM and a normalized tf-idf SVM with linear kernels.

3.2.2 Logistic Regression

Logistic Regression is a popular linear method for sparse text classification problems [15]. For text problems it is even a staple model for classification. We use the liblinear implementation present in the sci-kit learn package with L2-regularization. Like SVMs, this method employs a one-versus-all scheme and thus creates an extra classifier for each class. Additionally, we opt to include multinomial Logistic Regression via the Stochastic Average Gradient descent (SAG) solver[1]. We select both BoW and tf-idf variants, and normalize the input.

The method's inability to process all of the data during training time is problem. After some experimentation and deliberation, we decided to randomly sample 1,000,000 data points for cross-validation. Again, this introduces the same problem as with SVMs with regards to validity. However, we found that the logistic regression implementation has no trouble scaling to the complete validation data set during prediction-time.

3.2.3 kNN-limited

A proprietary kNN-based approach was developed within Atos. This method was implemented with some considerable constraints on vocabulary size and was tuned for speed (at prediction-time) over accuracy. While unfair for direct comparison because none of the other baselines considered here have

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

the same limits imposed on them, include the results from this method for sake of completeness.

3.2.4 Naive Bayes

Naive Bayes is interesting from a computational speed point-of-view, as the time to fit and predict on large amounts of data is almost trivial. Additionally, the Multinomial (Mult-NB) variants have documented success with text classification problems and an serve as a good baseline [40]. We do not impose constraints on the data sample size, allowing the supervised models to take full advantage of the data available.
As with the other models listed in this section, both BoW and tf-idf variants are trained.

3.3 Neural Networks

Artificial Neural Networks (ANNs) are a broad group of models that are specified by different architectures and their individual components, rather than algorithmic approaches with a number of parameters. As a high-level explanation, ANNs contain layers composed of neurons where operations on a input matrix (tensor) are performed. Each neuron in a network is able to individually tune to a subset of the data it is exposed to, and during back-propagation the error gradients are propagated back through the network in order to update the weights learned during training. An example of a very basic neural network consisting of an input, an output, and one hidden layer in between, is shown in Figure 3.1.

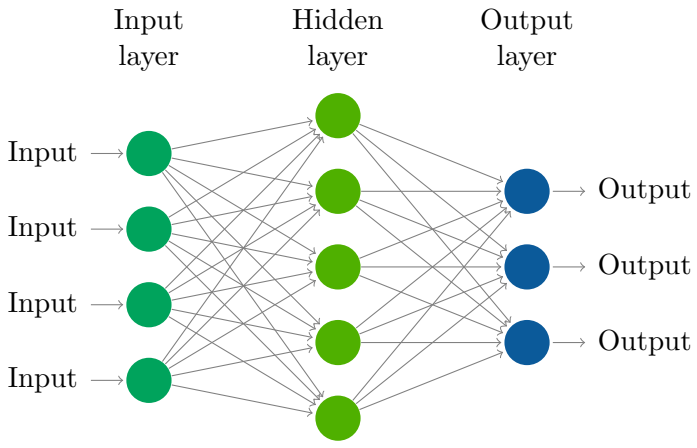


Figure 3.1: An example of an artificial neural network, consisting of one fully connected layer of five neurons in between four inputs and three output nodes.

Although they may seem like a dramatic change from the standard neural network model, convolutional neural networks are not all that different. CNNs combine neurons with a, often large number, of convolutional filters in order to obtain a activation map for the input. The neurons of the convolutional layer then respond to the input of the convolutions in order to produce their output. Specifically for text, convolutional layers are often used in different channels with each channel having a different filter size.

3.3.1 Building Blocks

Convolutional Neural Networks have a number of building blocks that are common among their architectures.

Embedding: words as vectors

In order to allow the network to make sense of words as input, we implement an embedding layer as a way of converting words in our dictionary to feature vectors based on the approach from [9]. This embedding layer is essentially a look-up table of size $m \cdot n$ where m is the dictionary size, and n is the length of the word embedding vector. For our work, we only include the top 20,000 words and include a word vector dimensionality n of 300. The rationale for limiting the number of words is to limit the memory use, and is further detailed in section 6.7. A useful property of this approach is that using a network can be allowed to fine-tune the weights in this embedding layer, allowing to pre-seed the embedding layer with previously learned word embeddings that can be updated by the model itself during training.

1D-Convolutional Layers

One-dimensional convolutional operations are at the core of CNNs and perform a convolutional operation on a sub-part of a sentence at a time. The convolution itself is done by picking a fixed-size filter[2] that is applied to the input to generate an one-dimensional activation map, as seen in Figure 3.2. The filter size as well as the stride can be defined. Common filter sizes for text are 2, 3, 4, and 5. Additionally, the number of filters is a hyper-parameter in these layers and corresponds to the dimensions of the output tensor. A sliding window approach is used to move across a sentence, continuously applying a convolutional kernel as it moves across a sentence. This method generates activation maps that are susceptible to words appearing next to each other regardless of their place in the input.

[2]Some sources in literature prefer kernel size, we opt here to use filter size.

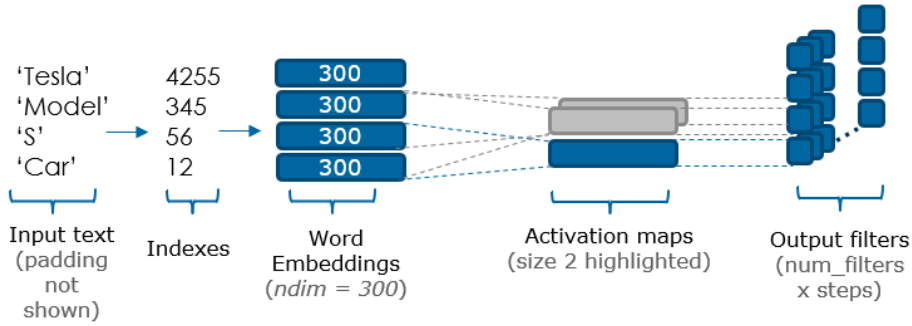


Figure 3.2: A abstraction of how the 1D convolutional operation operates on the word embeddings. The highlighted activation map targets two word embeddings (vectors) at once and outputs a number of filters. What is not shown is the padding of the input, which allows the convolutional window to move across the input text always returning the same size output tensor. For sake of completeness we also show the process of the conversion of the input text to indexes and word embeddings.

Fully-Connected Layers

Fully-connected or dense layers are relatively simple: they connect each of their neurons with both their input and their output. Fully-connected layers contain a pre-defined number of neurons that learn weights during training. In text CNNs, these layers are used to learn higher level representations based on the output of the feature-generating convolutional layers, often after the output has been flattened or pooled.

Pooling Operations

Pooling schemes have several effects in text CNNs. First and foremost, pooling typically serves to reduce the computational complexity, allowing the network to be larger. Max-over-time pooling is commonly used to select the most interesting features from each feature map [27]. Most notably, it introduces an invariance to sentence length. However, some authors (most notably [38, 24]) utilize a dynamic pooling scheme that pools input in chunks, in order to preserve some information about locality.

Batch Normalization

Batchnormalization is a relatively new technique and was introduced by Ioffe and Szegedy in a 2014 paper [21]. Batchnormalization normalizes the input per mini-batch for a layer, which smoothens the training process and provides some protection against overfitting. It functions as a way to accelerate and improve training of a neural network though it remains complex

to explain just why it works so well.

Dropout

Dropout randomly sets a neuron to zero and has a strong regularizing effect on the network, preventing overfitting and allowing the network to generalize better [52]. For our short text domain, the effect is much more pronounced than it is in computer vision networks and the network's performance can fall as a result. The probability of dropout setting a neuron to zero is given by p. Common choices include a p between 0.1 and 0.5. Dropout has somewhat fallen out of favour in recent years, as with deep learning the issue is generally fitting the network in the first place and learning representations. Other methods, such as batch-normalization, have also contributed to this. Without dropout overfitting remains an issue, but we argue it can be monitored with validation data and tools such as Google's Tensorboard[3] throughout the training process.

Output layer

The softmax function is used to output the probabilities per class of the network and contains n hidden units, where n is equal to the number of classes labels that appear in the data set. In our multiclass case, our output layer essentially performs a special case of multinominal logistic regression. Softmax itself includes a scaling process where all of the class probabilities are scaled to be within 0,...,1 and the sum of all probabilities is exactly 1. The largest probability is close to 1, while the probabilities that are much smaller will be close to zero.

In our case where we are only interested in the top probability as the predicted class label, we simply perform an $argmax$-function to obtain the highest probability.

3.3.2 CNN-Kim

Based on Kim's pivotal 2014 paper [27] we implement a convolutional neural network on the basis of the CNN-non-static model. The same hyperparameters are chosen as in the original paper, and we include the pretrained word2vec embeddings that were used in the original paper, publicly available and obtained via Google's archive[4].

Like in the original work, we allow the model to update its embedding layer. However, we do not replicate the embedding layer to have one individual

[3] https://github.com/tensorflow/tensorboard
[4] https://code.google.com/archive/p/word2vec/

layer for each filter size. This means that the network only has one embedding layer shared between the channels. It is possible that replicating the embedding layer multiple times allow the embeddings to be fine-tuned better. However, this means a tripling of the amount of parameters used, and to be tuned by, the embedding layer. We could also not find a clear answer in literature whether to replicate the embedding layer multiple times, and thus kept a single layer for all implementations, including our own method introduced in the next sub-section.

The network features channels which consist of a convolutional layer with a specified filter size on top of an embedding representation. This is then followed by max-over time pooling to select the most important word-level ngrams picked up by the filters. Like the paper, the model features a concatenation of three channels, with convolutional layers that have filter sizes of 3, 4, and 5 respectively. This is only passed on to a fully-connected layer with dropout and finally a fully-connected layer with a softmax activation to obtain class labels. Like the original paper, we choose Adadelta as the optimizer.

3.3.3 Simple CNN

As an extra baseline method and for the evaluation of our embedding strategies, we introduce a shallow simple CNN model consisting of a ReLU-activated convolutional layer of 256 hidden units with a filter size of 2 on top of an embedding layer. We apply no pooling, and instead include a dense layer with 1024 hidden units and ReLU along with dropout (p=0.5) in the architecture. This method leads to a very simple CNN that is relatively fast to train and performs reasonably well. The network is shown in Figure 3.3. The model is trained with the RMSprop optimizer with a batch size of 128.

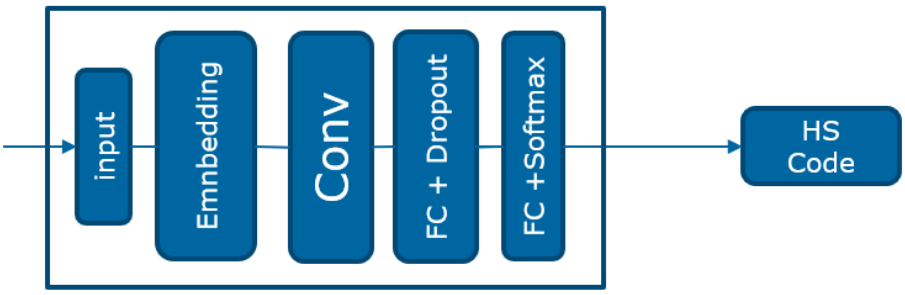


Figure 3.3: Architecture of the simple text CNN. We opted for this specific architecture because of the reduced training time. The network consists of a single convolutional layer with filter size 2, a dense layer, and dropout. Classification is done in the final softmax layer.

3.3.4 Own CNN

On the basis of work done by Kim [27] and Liu [38] we define a more complex CNN, illustrated in 3.4. This method includes the channel approach seen in their earlier work, and incorporates four channels of filter sizes 2, 3, 4 and 5. We choose to include the hidden bottleneck layer seen in Liu's XML-CNN to learn a more complex representation that can scale to the number of classes. This penultimate layer is also commonly seen in 2D and 3D-CNN models for computer vision. Unlike the XML-CNN however, we do not aim to perform multi-label classification: our HS labels can be assumed to be mutually exclusive.

An Embedding layer is constructed on the basis of the most frequent 20.000 tokens appearing in the data set. This is the same limitation as we impose on the other methods, but increasing the dictionary size would likely increase the model's performance at the cost of training time. We choose an embedding dimension of 300 that corresponds to our embeddings and allow a maximum sequence length of 16. Shorter texts are appended with zeros, and longer texts are cut off after the sixteenth word. Although the convolutional operations are agnostic to where word combinations happen, we choose to pad at the end of the sentence instead of prefixing our input if the size is shorter. For this implementation we allow the embedding layer to update its weights during training.

Each convolutional layer in the network contains 256 filters and uses ReLU as an activation function. We incorporate the use of batch normalization after each fully-connected and convolutional layer. The network as described is shown in 3.4. The network does not include the use of dropout, nor pooling schemes. Instead, we simply flatten the input after the convolutional layers to allow the fully-connected layers to take these as input. Lastly, the network includes a fully-connected layer with a softmax activation function as described in 3.3.1. We treat the highest probability returned as the model's class prediction.

3.4 Word Embeddings

In order to gauge the effect of different sources of embedding we use the Gensim[5] implementation of word2vec to train embeddings.

The original word2vec paper by Mikolov et al. introduced two modes: Skipgram and Continuous Bag-Of-Words (CBOW)[41]. The methods optimize different objectives in order to learn representations of words. In the

[5]https://radimrehurek.com/gensim/models/word2vec.html

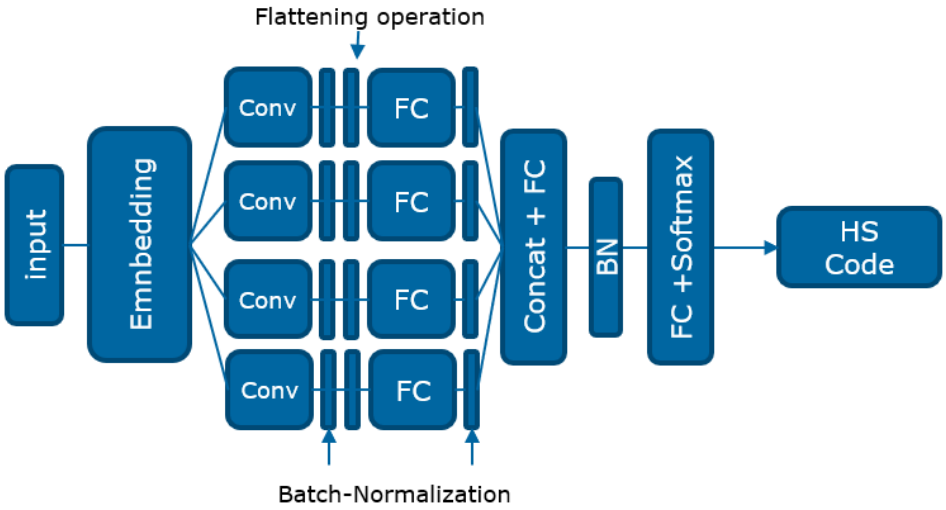


Figure 3.4: Our proposed network architecture. The input of the network is a list of tokens, which get translated to word vectors in the embedding layer. We use four convolutional layers (256 filters, filter sizes of 2, 3, 4 and 5)

negative-sampling Skipgram model the objective is to predict surrounding words given a target word [43]. In the CBOW architecture this is actually reversed: the goal can be described as predicting the current word given the context words surrounding it in the sentence it appears in [41]. Both of these methods produce weights taken from the hidden layer in the network, which provide a feature vector for each word in the vocabulary. It is worth noting that because the prediction task in CBOW only has one target it can be significantly faster.

3.4.1 Training Regime

For evaluating, we opt to train all embeddings with the Continuous-Bag-Of-Words training mode, where we use the mean value of the surrounding words as the basis of our embeddings. The reason for this is mostly practical: training CBOW embeddings is generally three to four times faster than skipgram. Any positive effect of better text data will be even more pronounced with skipgram using negative sampling, which we opt to select for our networks based on whichever CBOW model performs best.

In order to obtain word vectors we use the python library `gensim` by Radim Rehurek, as it includes a highly-optimized port of the original word2vec routines released by Mikolov in C.

Finding the best word vectors

For our task of finding word embeddings that work well with our domain data we first train the model on the original data (The Atos corpus for the Atos data set, and the Bills 2017 data for the Bills of Lading data set respectively, both are detailed in 4.1). In addition to the training regime described above, we choose a dimensionality of 300, bearing in mind that the quality of our embeddings might be greater with a higher dimensionality (Though it should be noted that Mikolov stressed this is not certain[6]).

We allow a context size window of 10, although most sentences in the data sets will be shorter than this. During training, we use the mean of the surrounding context words to establish the word vectors. We employ random downsampling ($p = 10^{-3}$) to mitigate the effect of high frequency words being dominant when calculating word vectors. Lastly, we use a corpus-wide minimal word count of ten: tokens appearing less than ten times are not considered.

After training on the original data, we train our embeddings for five epochs on the new corpus.

Best candidate training

Our best candidate we train under a skipgram objective with negative sampling. As before, we first train the model on the original data (The Atos corpus for the Atos data set, and the Bills 2017 data for the bills of lading data set). We again use a context size window of 10, a dimensionality of 300, and employ negative sampling at 5 noise words - the negative sampling exponent we use is 0.75 as in the original paper [43], as we are looking to sample more frequent words more often: we also chose this as we can expect the distributions in our data distribution to be similar to the corpus data. As with CBOW, we allow the model to train for five more epochs on each data source.

[6]See https://code.google.com/archive/p/word2vec/ under the section performance

Chapter 4

Data Methods

This section describes the data sets used in this work. We differentiate between the approaches for the two labelled data sets and the approaches to collect the natural language text data (corpora) used to obtain word embeddings.

4.1 Data Sets

4.1.1 Bills of Lading Data Set

The Bills of Lading (BoL) data set was obtained from the Engima Public data portal[1]. It is a collection of 38,084,118 Bills of Lading summaries from 2017, which are documents accompanying cargo containers. These are collected upon port entry by the U.S. Customs and Border Protection's Automated Manifest System (AMS) and are generally available to (paid) subscribers of commercial data vendor services. However, this set is available under a Creative Commons 4.0 Non-Commercial License, which allows for this data to be used for research. The data actually contains a wealth of other properties such as identifiers for the vessel carrying the cargo as well as the port of entry. **While we cannot verify the correctness of the data, we opt to use it to evaluate our approach**. We select only the text descriptions and HS-codes, to stay true to our goal of short text prediction. However, while the list of descriptions is complete we find that only 22.1% of all entries have a non-null HS-code. Upon inspection, it seems plausible that at least some of these shipments were entered under the wrong HS-code: for instance, we encountered an instance of oranges being imported under a vehicle chapter. We lack the means to manually correct these and inspect every item, and accept the erroneous classification as a potential issue stemming from using real-world data.
After removing empty data and invalid HS codes to obtain a complete data

[1]The data portal can be found at https://public.enigma.com/

we are left with a set of 7,641,625 descriptions and HS codes. This data set contains 99 labels at the HS-2 level and 1271 labels at the HS-4 level. Note that according to the Harmonized System some of the HS-2 chapters, such as code 77, are reserved for future use. Thus, we already encounter more labels than we can assume possible with compliant use of the standard.
As expected, we encounter a problem of skew towards short descriptions. While the mean length of a text description is surprisingly long (31 tokens) the median value is only 19. In fact, the most common text length is 2 tokens. Figure 4.2 illustrates the distribution of the tokens available per description of goods.
Where we do not encounter a short description, we often find that the text includes information that no longer describes the product itself but refers to the invoice, or simply includes multiple products in the same container. An example of this is given in 4.1: here we find that the text description contains a reference to an invoice.

PREIMPREGNATED DECORATIVE PAPER ON 10 PALLETS
PREIMPREGNATED DECORATIVE PAPER (FOIL) RELEVANT TO
INVOICE 45035866

Figure 4.1: An example of a text description, taken from the Bills of Lading summaries data set. Note that the quantity and the invoice are mentioned, and that the main description (preimpregnated decorative paper) is repeated.

Lastly, some of the product descriptions contain a HTS code, which is a ten-digit code that corresponds to a US-specific extension of the Harmonized system. This could be a serious problem for the validity of our approach, as any classifier could simply learn to associate these with a HS label. In order to mitigate leakage via these we remove any word that only contains digits and is longer than six digits.

4.2 Atos Data Set

The Atos data set was collected in collaboration with a customs authority Atos has partnered with. The data set includes 5,457,325 descriptions of goods and their corresponding HS label, up to a HS-8 definition. These descriptions were collected from electronic administrative intake forms as they were submitted to the customs authority between 2015 and 2017. The HS label data is assumed to have been corrected by the customs authority and the descriptions are in general, of high quality. In total the data set contains 5061 unique HS-8 labels, which are aggregated into 96 HS-2 labels and 1207 HS-4 labels. The number of unique goods descriptions in the data set is quite large: 3,108,080 (57%) text descriptions appear only once in the

data set. The average "sentence" length is quite small: the mean number of tokens encountered per description is 8.05, and the median value observed is 7. The most frequent token count however, is just three tokens, with the next largest group being descriptions that are just two tokens, as illustrated by 4.3.

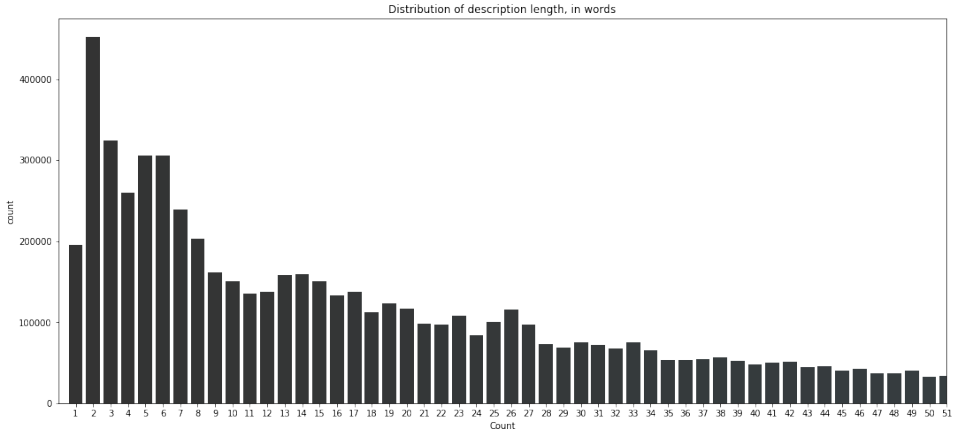


Figure 4.2: The distribution of description lengths in the Bills of Lading from 2017 data set. The counts shown are the number of words in the descriptions. The counts notable peak at 2 tokens.

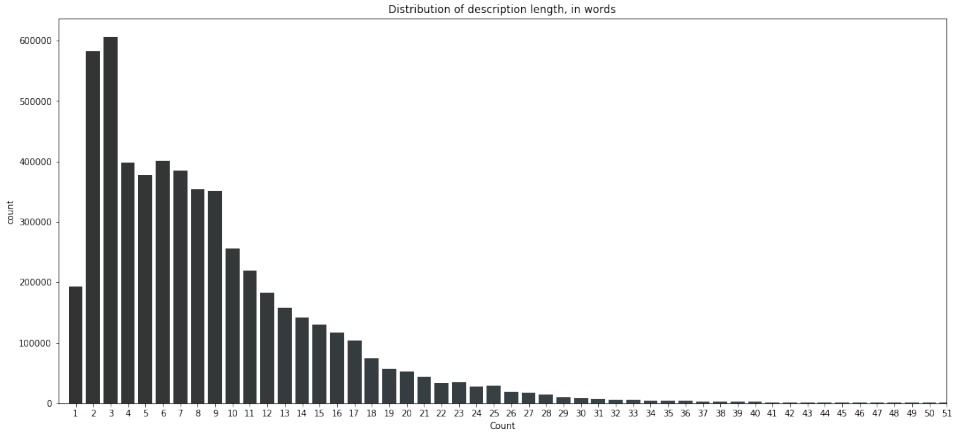


Figure 4.3: The distribution of description lengths in the Atos data set. The counts shown are the number of words in the descriptions.

There is a significant imbalance in the class distribution, with the smallest class, 43 (Furskins and artificial fur; manufactures thereof), being composed of only 47 items while the largest class, 87 (Vehicles other than railway

or tramway rolling-stock, and parts and accessories thereof), accounts for 953,291 data points (17.5% of all data). As such, there is not just a skewed distribution with regards to overall description length, but also in the class frequency distribution.

Examples

The Atos data set is a company-specific data set and as such will not be released to the public domain with this thesis. We have therefore opted to include comprehensive lists of the amount of data points per HS-2 code and the descriptions of each HS-2 chapter to the appendix to this thesis.

Table 4.1 directly illustrates some of the difficulties encountered with this data set. There are multiple items (item number 2, 3, and 4) that are essentially spare parts for some kind of machine: yet, these fall under three distinct chapters. Another salient feature of is the brevity of these text samples: an item such as *"handbags"* is just one word. *"Spice mixed"* or *"new drain"* is similarly just two tokens. This serves to further illustrate the low information that can be gathered from these texts. Thirdly, some of these texts include potential valuable keywords such as *"New"* and *"In Transit"*.

HS-6 code	Description
271019	GASOIL IN TRANSIT TO SOUTH SUDAN.TRUCK
842123	NEW ATLAS COPCO AIR COMPRESSOR SPARE PARTS : PREVENTIVE MAINTENANCE KIT / SERVICE KIT : ORIGIN BELGIUM
854370	NEW DRAIN
340319	NEW OIL PISTON
420222	HANDBAGS
091091	SPICE MIXED
121190	BASIL SEEDS (TAKMARIYA)

Table 4.1: Some descriptions and their corresponding HS-6 code from the Atos data set. The descriptions are representative in the shortness of descriptions encountered in the data set.

4.2.1 Preparation and Discussion

Pre-Processing

For the data sets under consideration we perform a number of steps to pre-process the data. By means of a regular expression, we first obtain a format of one description per line. We then replace diacritics with their unaccented form and convert the text to lowercase UTF-8.
We preserve hyphens, but otherwise have a rather strict removal policy: we only keep alpha-numeric content, including spaces. However, extra white space is also removed, as words will be tokenized using the remaining spaces. However, we do not correct for spelling mistakes (such as dinning instead of dining) which are frequent with our human-entered data. We do also not remove stop words, or employ a stemming or lemmatization strategy. Lastly, we do not group common phrases together to make them appear in the data as one token (such as "new_york" instead of "new york"). We also do not perform any part-of-speech (POS) tagging, syntactic dependency parsing, or named entity recognition (NER).

Summary and Discussion

We have been unable to find a curated data set in our domain of harmonized code classification to establish a benchmark. On a high level, the data sets we have found meet our goal of short text description. Of the two, the public Bills of Lading 2017 data set contains longer descriptions and is noisier. The Atos data set contains many shorter descriptions and can be assumed to be corrected, but it is not in the public domain. In order to allow for comparison, we will compare classification approaches on both data sets.
Table 4.2 shows the number of data points in both data sets, along with the number of unique labels encountered at both levels of distinction.

Data Set	Number of Samples	HS-2 Labels	HS-4 Labels
Bills of Lading Summaries	7,641,625	99	1271
Atos	5,457,325	96	1207

Table 4.2: Description of the data sets used in this project.

4.3 Data Collection for Word Embeddings

We collected additional data from several sources that we deemed likely to have product information. The first corpus[2] is based on ontology data collected from Product Ontology[3], which itself is based on previously identified products and services on Wikipedia [19]. We used these to target pages on Wikipedia which we subsequently scraped. A second corpus comes from DBpedia itself, where we directly query for abstract texts based on ontology types. Lastly, we created corpora from the data sets used in this thesis as well as a collection of text from bills of lading documents from 2016 and 2017, which includes the data set of 2017 that appeared earlier in this chapter.

4.3.1 The Product Ontology Corpus

RDF Data was collected from a public-facing API on productontology.org, a previous collected work of 300k product definitions in RDF-format which extends the GoodRelations schema [19]. The Product Ontology work is of particular interest, because it contains only items that are classified as products, and it is based on Wikipedia entries. We theorized that we could use this data to pin-point pages on Wikipedia that included items that were likely to be imported. We aim to scrape the summary contents of these Wikipedia pages with the `wikipedia` library for Python to create a corpus we could train a word2vec model with.

However, the public Product Ontology API only exposes items that have been requested as pages at least once, which gave us a data set composed of only 23,731 items and not the 300,000 items that were involved in the work. As some of the items on in this data set currently refer to disambiguation pages on Wikipedia or pages that have simply been removed since, we disregarded these while scraping from Wikipedia. As a result, only 20,578 Wikipedia pages could be collected for their summaries. We split this collection into sentences and clean the data, replacing diacritics with their unaccented form as we go. Out of this, we construct a corpus consisting of 127,260 sentences with 2,742,116 words to train on, for an mean sentence length of about 21.5 words.

[2] We adopt nomenclature from corpus linguistics and refer to our all-text data sets as corpora.

[3] The Product Ontology work includes a public-facing API, at http://www.productontology.org/

4.3.2 DBpedia as a Corpus

A second corpus was collected by querying the public DBpedia SNORQL explorer[4]. We first identified ontology classes that were likely to hold product information, such as ChemicalSubstance, Food, and Device. A list of our target ontologies and the number of objects retrieved per ontology type is shown in table 4.3.

Type	Number of objects included
Animal	229,955
Device	11,957
Company	107,357
ChemicalSubstance	18,505
Biomolecule	7,335
Food	11,358
Product[5]	55,911
Plant	62,277
MeanOfTransportation	55,325

Table 4.3: Ontology types and the number of objects retrieved per type. We filter to only retrieve items that have an abstract in the English language defined.

With SPARQL queries to retrieve these in a JSON structured format we obtained these objects only if the item had a `DataTypeProperty: has abstract` defined. In fact, we are only interested in these because they contained full-text paragraphs describing these products. We also select only those items which are defined to be in English. We employ the same cleaning process as with the Product Ontology corpus, splitting the corpus into sentences and preserving only alphanumeric content. The resulting set contains 2,287,513 sentences and 40,916,158 words, yielding an average sentence length of 17.9 words.
The SPARQL queries corresponding to these can be found in the online appendix to this document.

4.3.3 Corpus from Bills of Lading Data

Finally, we have a large repository of Bills of Lading data available through the same public data repository as the bills data set. We merge the data

[4]Found at http://dbpedia.org/snorql/

[5]Product ontology data was obtained by querying for the schema.org product definition of the term Product, which is actually again based on the GoodRelations definition by Martin Hepp, as published in [19].

from 2016 and 2017 and filter for the text descriptions. This means that for the corpus construction we also include text descriptions we originally discarded for the 2017 data set because they had no HS-label associated. This grants us a collection of 72,069,519 documents. This 'Bills of Lading' corpus contains 1,868,260,039 tokens. We process this text in a similar way as the other corpora.

4.3.4 All Corpora Combined

We also construct a corpus out of all our corpora together to give us a corpus of 1,958,887,695 words. The word and sentence counts are shown in table 4.4. This table also serves to summarize the number of words and sentences in each of the corpus.

Set	Method of Collection	Words	Sentences
Atos Data	All text in Atos data set	46,969,382	5,457,241
Dbpedia Abstracts	Queries	40,916,158	2,287,513
ProductOntology	Web scraping / Wikipedia	2,742,116	127,260
BoL 2016	Open Data	885,208,564	33,985,497
BoL 2017	Open Data	983,051,475	38,084,022
Sum[6]		*1,958,887,695*	*79,941,533*

Table 4.4: The total number of words and sentences in the corpora along with a brief description of how they were obtained.

4.4 Corpus Preparation

For the corpora we employ a similar strategy as the data sets, with only a few differences. We convert and replace diacritics and punctuation, but split sentences to obtain a format of one sentence per line. This allows them to be used in conjunction with the Gensim Python package's PathLineSentences[7] function to create a generator object to read and train with our large corpus files. We found that this streaming approach was an absolute necessity to not run out of memory.

[6]The sum here is equal to the size of the corpus 'All corpora combined' as it appears in the results section.

[7]https://radimrehurek.com/gensim/models/word2vec.html

Chapter 5

Results

We here turn to answering our main objectives: what the influence is of our embedding approach, and the results of our classification of Harmonized System codes on the HS-2 and HS-4 level. For the latter, we list scores for both levels separately. As described fully in the Method section, we evaluate on the basis of a previously split-off 20 pecent of the data, and for models that cope badly with this amount of data points we use a cross-validation process. The test set portion of the data was not used for development.

We do not include confidence intervals for the 95 percent confidence level. Due to the large test set size, the confidence intervals we found are on the order of ± 0.002 for kNN-Atos and the SVM models, and ± 0.0007 for the rest of our models. For the latter, this effectively means that when rounded, this can be construed as a blanket score of ± 0.001 on all reported scores.

5.1 Constraints

We limit the tokens space to the top 20,000 most frequent words for all approaches. We believe that there is no doubt this will impact some models more than others, and it will be discussed further in section 6.7.

5.2 Domain-Specific Embeddings Training

Based on the shallow simple CNN model outlined in section 3.3.3, we train and evaluate several sources of embeddings. For each embedding we test variants where fine-tuning on the embedding layer is allowed (trainable) and in which it is kept static. Fine-tuning in this sense allows the model to update the weights of its embeddings layer. We restrict the token space to the top 20,000 tokens in the input data set and limit the maximum length of a sentence to 20 tokens. Shorter descriptions are padded with zero space. Models are trained until convergence. The F1-score results on the test set

of the Atos data set are shown in Table 5.1.

Embeddings source	Weights	F1-score	recall	precision
Own Data (baseline)	static	0.81	0.81	0.82
Own Data (baseline)	trainable	0.82	0.83	0.83
ProductOntology	static	0.83	0.83	0.83
ProductOntology	trainable	0.84	**0.84**	0.84
DBpedia	static	0.82	0.82	0.83
DBpedia	trainable	0.83	0.83	0.84
All corpora combined	static	0.83	0.83	0.84
All corpora combined	trainable	**0.85**	**0.84**	**0.85**

Table 5.1: Results of models trained with different sources of domain-specific embeddings using the Continuous Bag of Words (CBOW) scheme. The F1-score on the Atos data set is reported.

We select the best candidate of the domain-specific embeddings and use the corpora it was trained on to train a negative-sampling skip-gram variant[1].

5.3 HS-2 Classification

For each of the classifiers defined in 3, we train and evaluate based on the performance over a train-test split. In the case of classifiers such as SVMs and Logistic Regression, we evaluate on the basis of a cross-validation process with a randomized sub-sample of the data. For the simple CNN and the Own Method CNN, we use the word embeddings trained on all corpora. For the CNN based on Kim 2014 [27], we use the publicly accessible word2vec Google News embeddings trained on 100 billion words as introduced in [43]. Models are trained until convergence, as we report the F1-score. The results are shown in Table 5.2.

5.4 HS-4 Classification

We approach the problem of HS-4 classification similarly to HS-2 classification. However, despite significant troubleshooting we find that most

[1]An exhaustive comparison between word2vec variants is not possible due to time constraints. We acknowledge that the statement that any gains that appear under a CBOW objective might not necessarily hold for skip-gram. However, we informally tested skip-gram variants based on the baseline, the ProductOntology corpus, and the all corpora combined corpus and the results were in line with what we report in section 5.1

Model	Method	BoL	Atos
MNB-tfidf	Cross-validation	0.807	0.820
MNB-BoW	Cross-validation	0.775	0.818
SVM-tfidf	Cross-validation	0.815	0.812
SVM-BoW	Cross-validation	0.746	0.788
LR-tfidf	Cross-validation	0.857	0.853
LR-BoW	Cross-validation	0.802	0.834
MLR-tfidf	Cross-validation	0.859	0.857
MLR-BoW	Cross-validation	0.799	0.841
kNN-Atos	Train-test-split	-	0.761
CNN (simple)	Train-test split	0.827	0.859
CNN (Kim + Google word2vec)	Train-test split	0.8742	0.849
CNN (Own Method)	Train-test split	**0.923**	**0.902**

Table 5.2: Results of classification on the HS-2 (chapter) level. We report the F1-score of each method. Where noted the method of evaluation involved cross-validation (with number of folds equal to 5), otherwise the results reported are on the same train test split described in the previous section. It is important to note that all of the SVMs and all the logistic regression models involved a random sampling strategy because these methods were unable to process all of the training data.

implementations are no longer able to scale to both the amount of labels and the amount of data. The Bills of Lading data set contains 1271 classes at this level and the Atos data set contains 1207. Our approach for the HS-4 level thus only includes three CNN variants. We report accuracy at the HS-4 level, as we encounter problems with weighting schemes for macro F1-scores with classes with no predicted samples. Like the HS-2 prediction, we use the all corpora combined embeddings for the simple CNN and the Own Method CNN, and the public word2vec embeddings for Yoon Kim's CNN. We allow each model to update its embedding layer's weights. The results are shown in Table 5.3.

Experimental set-up

All experiments were performed on a virtual machine with eight CPUs (Intel Xeon E5-2673 v3, clocked at 2.40GHz) and 56 GB ram available. Where reported, processing speed refers to the time taken on this machine.

Model	Method	Bills data set	Atos Dataset
CNN (simple)	Train-test split	0.658	0.726
CNN (Kim + Google word2vec)	Train-test split	0.783	0.763
CNN (Own Method)	Train-test split	**0.880**	**0.841**

Table 5.3: Results of our approaches on the HS-4 level. We report accuracy across all predicted samples at the HS-4 level.

Chapter 6

Discussion

6.1 Web Scraping and Corpus Construction

Through web scraping we were able to collect a broad collection of data for a corpus. However, our approach was naive, as we blanket-retrieved text we hoped would contain enough context. As such, we still suffered from the problem of not being able to know for sure whether we were actually collecting the correct data off the web and the scale of our data collection ruled comprehensive inspection impossible.
However, we still were able to collect tens of thousands of object descriptions from the Product Ontology API and hundreds of thousands of object descriptions from DBpedia. This data significantly improved the performance of our models compared to those that just had Word2Vec models trained on the data itself.
We expect that we could use the same process that we followed with the Product Ontology scraping to target pages associated with entries on DBpedia, to obtain more comprehensive corpora on products, services, and materials.

6.2 Embeddings

The embeddings we obtained from each corpus, and particularly the combination of all sources, proved useful in increasing the performance with regards to the baseline. It seems that the precision is slightly higher than the recall in almost of our evaluations, but overall that the networks try to find a good balance between recall and precision based on the F1-score metric. The biggest increase was seen when using the embeddings trained on all corpora combined, and these were then used for training a negative-sampling skipgram model for HS-2 and HS-4 classification. Surprisingly, despite their limited size (41M words versus 1,959M words) the embeddings learnt from the corresponding entries on Product Ontology their Wikipedia

pages proved to good contenders. We feel this suggests that longer, natural language texts as they appear on Wikipedia, are excellent proxies for short text descriptions.

Although the intent was never to directly compare against open-source pre-trained embeddings (again, mostly due to time constraints), we found that for our simple network using our best candidate embeddings performed slightly better than using pre-trained embeddings, and that for more complex networks- our own and that based on Yoon Kim's work - the effect was much more pronounced. We believe this may be due to the fact that higher-level domain-specific word groupings are more likely to appear in domain specific texts than in general embeddings, and that models trained on this are then better able to make sense of their short input later on.

6.3 Word Embeddings Interpretability

Interpreting what is actually learned with word embeddings remains an open subject and seems to vary significantly between runs on the same data [49, 6]. However, word embeddings are essentially word representations in a high dimensional space, and while the representation may vary each time a word embedding model is (re-)trained, any method that merely consumes these as features will have to be re-trained on them regardless.

To illustrate this, a list of the ten closest words to `spice` in the plain Atos data set's Word2Vec model is shown in Table 6.1. While the vectors representing these terms will change every time a Word2Vec-model is re-trained, the cosine distances (the ranked similarities) should stay relatively unchanged. Interestingly, in the excerpt the term `freddy` here refers to a spice company from South Africa. Other than that all of the items listed seem to directly refer to spice and herbal items.

Term	Cosine distance
‘spices’	0.7483088970184326
‘masala’	0.7138966321945190
‘jeera’	0.6665996909141540
‘curry’	0.6586883664131165
‘garam’	0.6336268186569214
‘seasoning’	0.6325685977935791
‘paprika’	0.6321157217025757
‘herb’	0.6250514984130859
‘freddy’	0.6206517815589905
‘turmeric’	0.6175587177276611

Table 6.1: A list of terms ranked closest to ‘spice’, ordered by cosine distance. Retrieved from a Word2Vec model trained on the Atos data set.

We can also employ t-Distributed Stochastic Neighbor Embedding [39] to provide a 2-dimensional mapping of words from the same Word2Vec model. This illustrates that words that appear in the same context appear in the same region. An example of this is shown in Figure 6.1, where the words around the term ‘spade’ are visualized. Although the plot is not centered, we encounter many domain terms such as slasher (refers to a long machete-like tool and not the movie genre), panga (refers to another machete-like tool, and not for instance one of the many fish species with that name), and hoe (refers to the ground-moving tool) in the direct vicinity.

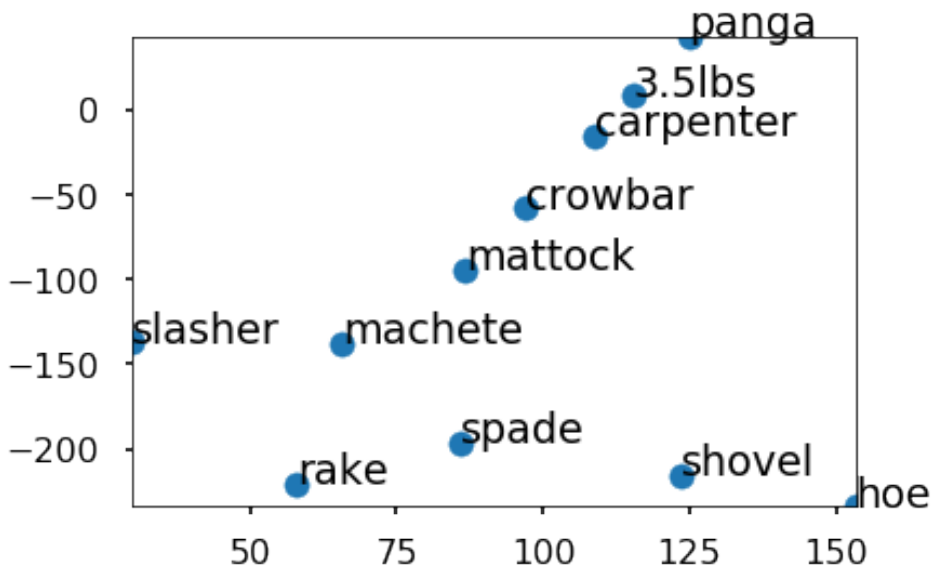


Figure 6.1: A t-SNE mapping visualized showing the ten closest terms to the term 'spade' in a learned representation from the Atos data set.

6.4 HS-2 Classification Results

For HS-2 prediction we were able to test a wide range of methods, including some classical approaches. Our CNN ('own method') yielded the best results of all methods tried. We obtained a F1-score of 0.902 on the Atos data set and a F1-score of 0.923 on the BoL 2017 data set. We believe the choice of a filter with size 2 (something not used in CNN-Kim) allowed the network to capture more short snippets that were relevant for prediction. This is somewhat supported by the good baseline performance of the simple CNN model, which focused on that particular filter size alone. However, we did not incorporate a version of CNN-Kim with a channel with filter size 2 to verify this.

Under the constraints we imposed, the logistic regression models based on tf-idf features worked well out the classical approaches, even competitive to implementations of our simple CNN and CNN-Kim. The main reason why the classical approaches were not able to achieve higher performance was due to the fact that they all coped with either trouble scaling to the amount of training samples or the number of classes. Inherently, a text classification problem is characterized by a sprawling sparse feature space, and this greatly limits the effectiveness of SVMs and Logistic regression. In every instance where a direct comparison was possible, the tf-idf methods outperformed their bag-of-words peers. We are uncertain if this would translate to CNNs,

such that perhaps that our CNNs could also find merit in weighting word embeddings using a tf-idf scheme.

6.5 HS-4 Classification Results

For HS-4 classification we employed three convolutional neural network architectures. For both data sets, our own method completely outperformed the other models. We believe this success can be largely attributed to the inclusion of extra fully-connected layers (similar to 'hidden bottleneck' layers [38]) that allowed the network to learn finer-grained definitions of each label at the HS-4 level; perhaps this was only a modest effect for HS-2 classification. Secondly, the separate channel with filter size 2 here might also allow to capture simple 2-grams. Lastly, our model is different since we make liberal use of Batch-Normalization, which we did not encounter in previous work.

Interestingly, the Simple CNN we defined no longer performed better than the CNN-Kim model on the Atos data set. Perhaps this is due to an inability to capture the higher level n-grams that define the more complex HS-4 classes, as it is known that the verbosity of descriptions increases as the resolution increases.

6.6 Applicability to HS-6 and Beyond

We expect that the models proposed here can also be used for classification at the HS-6 level, and even beyond. However, we were unable to do so within the scope of this thesis. Several challenges remain at this level, as there often are multiple completely valid classifications for a short text and there simply is not enough context to determine which one is the most accurate. For instance, a description of 'printer' still has many valid choices, such as labels for ink-jet printers, printing machines, laser printers and so on. The Harmonized System assumes only one correct classification. It is likely our method can be improved by involving other data that might be gathered on the goods, such as x-ray scans, physical dimensions, whether it is stored in a refrigerated container, and other information present on the declaration forms.

6.7 Constraints and Limitations

Although the relatively high F1-scores and accuracy may make it appear otherwise, this study is highly constrained.

6.7.1 Dictionary Limitations

First and foremost, we limit the amount of unique words in our data to only consider the top 20,000 words. This was done mostly due to computational constraints: we encountered memory limits on the objects and tensors we created and we could not always work around these by means of batching or streaming. Another reason can be found in that the experiments were done on a set of CPUs, which are much less optimized for matrix computations - which dominate convolutional neural networks. On the set-up we used (briefly described in 5.4), we found training a single CNN could easily take more than a couple of days, regardless of data set.

We limited the input for the simple-CNN to 20 tokens, and for the own method-CNN we even reduced this to 16. We believe performance is limited by this truncation, particularly on the longer texts that appear in the Bills data set. Though by far this is enough for most of our sentences; particularly because the relevant information is often placed at the start of a description.

In order to enforce the same limits on our baseline classifiers, we also selected the top 20,000 for the BoW and tf/idf features. For some classifiers this served to eliminate poorer features, increasing the performance. Others, most notably logistic regression, saw a reduction in performance as a result.

6.7.2 Word2Vec Limitations

With regards to Word2Vec, we limited the number of epochs, dimensions, minimal word count across the corpus, and the context word window. Although we could not always find a clear answer in literature, we suspect these can be increased at the cost of training time to provide higher-quality embeddings.

6.7.3 Possibility of Fraud and Smuggling

On the topic of data quality, we implicitly accepted that there are possible fraudulent entries present in our data. For instance, on numerous occasions we encountered entire cars being labelled as spare parts - which carry a lower tariff. Training a classifier on this is problematic, as it will tend to learn these cases. We lack the means of verifying and correcting this and had to accept the bias.

6.8 Implementation Notes

In our implementations we observed some other phenomena that are perhaps more of interest to deep learning practitioners in industry working on

product classification than others. This section lists a small number of them. As the HS system is hierarchical, it is possible to train a model for a complex level, say HS-4 or HS-6, and retroactively classify for HS-2 by taking the first two digits of the model's prediction. However, we found that this method does not lead to an increase in performance when compared to a specific HS-2 model of the same CNN architecture, seed, and configuration. In fact, a purpose-trained HS-2 classifier always performed several percent-points better. We believe it is plausible that an ensembled approach of classifiers working at different HS levels increases classification accuracy.

We were also able to apply our proposed method in a recommendation-like setting, in which we counted success as having the true class (the relevant result) in our top-n. Informally we trialed this on the Atos data set. We obtained an accuracy of 93 percent at the top-3 level, 95 percent at the top-5 level and 97 at the top-10 level. As for this thesis we were only interested in the top result, we chose to not include this in our result section and abandoned this direction. Furthermore, we lacked the means to include this for every model and data set combination we included here.

Scraping of wikipedia was done using the python `wikipedia` package, which automatically spaces out requests to not strain web resources. Unsurprisingly given our number of requests, we found this package to be rather slow.

All experiments in this thesis were done on a CPU set-up. However, in a brief trial with a NVIDIA Tesla k80 GPU, we were able to speed up our CNN training by a factor of 10.

Chapter 7

Conclusions

In this thesis we set out to tackle the problem of HS-code prediction at the HS-2 (chapter) and HS-4 (heading) level on two real-world data sets that were defined by their large size and short text descriptions. We first learned domain-specific word vectors from text on the web, and based on that developed an approach with convolutional neural networks (CNNs) that significantly outperformed baseline methods. On the HS-2 level, our CNN was able to classify with a f1-score of 0.9 and 0.92. On the HS-4 level we evaluated based on accuracy and scored 0.84 and 0.89 on both data sets respectively. We find strong indications that this performance is achieved due to our neural network architecture, which includes the learning of small filters, batch-normalization, and the inclusion of fully-connected layers which use the convolutional layers as feature generators. Secondly, for the purpose of learning embeddings we demonstrated that we could obtain custom domain-specific corpora by using information from Wikipedia, DBpedia, and open data portals.

7.1 Future Work

For future work on this problem we would like to work on character-level convolutional neural networks, as the relative brevity of these texts in general is challenging for the more conventional word embedding models. Character level CNNs show promise and would not need word embeddings as input data.

Our architecture can also be further fine-tuned and might be needlessly complex in terms of filters and hidden units, while it could perhaps also benefit from a carefully-chosen pooling scheme. We did not include a pooling scheme as we were unable to achieve any boost in classification performance. However, because these reduce the amount of features at their step, pooling operations could speed up the training of our network.

At the time of writing things are moving extremely fast for natural language processing. Work with attention, or transformer mechanisms, such as in [53, 13] indicates strong success in question answering and other NLP tasks. It could be worthwhile to adapt a trained model such as BERT for the problem of HS code classification - in a novel application of a complex work.

An interesting improvement that we acknowledge but could not pursue in this project was the method proposed by Arora et al. in 2016, where they combined word embeddings with a tf/idf weighting scheme and modified by Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) [3]. Recent work indicates that this method is still hard to beat for short text documents [48]. This could perhaps improve our classical baselines - and interestingly be incorporated in our CNNs.

7.1.1 Vectors for Improvement: Improving These Results

The existing network could be improved by increasing the number of words considered in the model. In our work, we limited this to 20,000. Increasing the dictionary size seems like a relative easy path to increase performance, as the cost of complexity and time. Improvements on the architecture may also include more channels as well as ensemble strategies.

7.1.2 Acknowledgements

This work was made possible by an internship position at the department of Data Analytics and AI of Atos Netherlands. In addition, access to the Bills of Lading data set[1] was obtained under a Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license[2].

7.2 Online Appendix

The online appendix to this work, which includes example code of the network proposed here, can be found at https://github.com/jeffluppes/textcnnthesis.

[1]The Bills of Lading data was obtained from https://public.enigma.com/datasets/bill-of-lading-summary-2017/0293cd20-8580-4d30-b173-2ac27952b74b

[2]https://creativecommons.org/licenses/by-nc/4.0/

Bibliography

[1] Omar Abdelwahab and Adel Elmaghraby. 2016. UofL at SemEval-2016 Task 4: Multi domain word2vec for Twitter sentiment classification. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*. 164–170.

[2] Benjamin Adams and Grant McKenzie. 2018. Crowdsourcing the character of a place: Character-level convolutional networks for multilingual geographic text classification. *Transactions in GIS* 22, 2 (2018), 394–408.

[3] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. (2016).

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[6] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*. 4349–4357.

[7] Hon-Ki Chan, Huarong Zhang, Feng Yang, and Gunter Fischer. 2015. Improve customs systems to monitor global wildlife trade. *Science* 348, 6232 (2015), 291–292.

[8] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing

(almost) from scratch. *Journal of machine learning research* 12, Aug (2011), 2493–2537.

[10] Alexis Conneau, Holger Schwenk, Yann LeCun, and Löc Barrault. 2017. Very deep convolutional networks for text classification. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*. Association for Computational Linguistics (ACL), 1107–1116.

[11] W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*. Vol. 520. Addison-Wesley Reading.

[12] Rodrigo de Abreu Batista, Daniela DS Bagatini, and Rejane Frozza. 2018. Classificação Automática de Códigos NCM Utilizando o Algoritmo Naïve Bayes. *iSys-Revista Brasileira de Sistemas de Informação* 11, 2 (2018), 4–29.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[14] Liya Ding, ZhenZhen Fan, and DongLiang Chen. 2015. Auto-categorization of HS code using background net approach. *Procedia Computer Science* 60 (2015), 1462–1471.

[15] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research* 9, Aug (2008), 1871–1874.

[16] Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57 (2016), 345–420.

[17] Office of the Auditor General of Canada Government of Canada. 2010. 2010 Fall Report of the Auditor General of Canada. `http://www.oag-bvg.gc.ca/internet/English/parl_oag_201010_08_e_34291.html#hd4b`

[18] Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

[19] Martin Hepp. 2008. Goodrelations: An ontology for describing products and services offers on the web. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 329–346.

[20] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 328–339.

[21] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*. 448–456.

[22] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding Convolutional Neural Networks for Text Classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 56–65.

[23] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, 137–142.

[24] Rie Johnson and Tong Zhang. 2015. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 103–112.

[25] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 655–665.

[26] Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 1411–1420.

[27] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1746–1751.

[28] Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1329–1333.

[29] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

[30] Hoa T Le, Christophe Cerisara, and Alexandre Denis. 2018. Do convolutional networks need to be deep for text classification?. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

[31] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.

[32] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.

[33] Guo Li and Na Li. 2019. Customs classification for cross-border e-commerce based on text-image adaptive convolutional neural network. *Electronic Commerce Research* (2019), 1–22.

[34] Yichen Li, Arvind Tripathi, and Ananth Srinivasan. 2016. Challenges in Short Text Classification: The Case of Online Auction Disclosure.. In *MCIS*. 18.

[35] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. 2015. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 136–140.

[36] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).

[37] Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1299–1304.

[38] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 115–124.

[39] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[40] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[41] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. `http://arxiv.org/abs/1301.3781`

[42] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association.*

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.

[44] David L Olson and Dursun Delen. 2008. *Advanced data mining techniques.* Springer Science & Business Media.

[45] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).* 1532–1543.

[46] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* 2227–2237.

[47] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language%20understanding%20paper.pdf* (2018).

[48] Craig W Schmidt. 2019. Improving a tf-idf weighted document vector embedding. *arXiv preprint arXiv:1902.09875* (2019).

[49] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* 298–307.

[50] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34, 1 (2002), 1–47.

[51] Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 669–674.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[54] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification.. In *IJCAI*. 2915–2921.

[55] Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 90–94.

[56] Carsten Weerth. 2008. Basic Principles of Customs Classifications under the Harmonized System. *Global Trade & Cust. J.* 3 (2008), 61.

[57] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[58] Hsiang-Fu Yu, Chia-Hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih-Jen Lin. 2012. Product title classification versus text classification. *Csie. Ntu. Edu. Tw* (2012), 1–25.

[59] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

[60] Ye Zhang and Byron Wallace. 2017. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 253–263.

Appendix A

Appendices

A.1 Description of the Atos Data Set

The following table lists the Atos data set in terms of samples and number of labels, broken down per HS-2 chapter.

HS-2 code	Number of labels	Samples
1	27	2447
2	59	10380
3	82	21093
4	34	15823
5	15	1047
6	16	502537
7	66	123097
8	65	50256
9	32	204839
10	22	22440
11	28	9268
12	45	13418
13	9	1200
14	4	5151
15	47	23052
16	26	5652
17	23	23864
18	10	18721
19	23	32205
20	52	33979
21	20	38058
22	26	98401
23	20	10179
24	11	6509
25	64	37371

26	28	3673
27	64	133588
28	160	24551
29	295	24173
30	32	73093
31	23	6279
32	47	38690
33	29	56629
34	24	40265
35	15	11923
36	8	1488
37	29	3868
38	92	48133
39	153	220380
40	85	119650
41	29	10133
42	20	27650
43	6	47
44	76	37025
45	7	116
46	11	2211
47	15	465
48	110	97768
49	22	36272
50	6	113
51	27	498
52	104	12019
53	21	1905
54	66	9483
55	99	11953
56	31	10300
57	21	9307
58	36	6661
59	24	5240
60	38	4547
61	106	39803
62	115	58914
63	55	88052
64	26	35579
65	10	8706
66	6	4575
67	8	4101
68	50	20973
69	29	74290
70	68	53308

71	44	7421
72	162	57334
73	131	186664
74	52	7678
75	16	251
76	39	27514
77	0	0
78	8	295
79	9	1688
80	5	120
81	26	249
82	64	75579
83	43	56895
84	520	511069
85	273	404648
86	20	10607
87	106	953291
88	14	8777
89	18	974
90	146	100667
91	44	6580
92	17	4441
93	18	801
94	44	152899
95	31	34652
96	51	54467
97	7	2296
98	0	0
99	0	0