Hierarchical multi-class classification of imbalanced product descriptions using Support Vector Machines and BERT

Meulen, M.J.E. van der

Hierarchical multi-class classification of imbalanced product descriptions using Support Vector Machines and BERT

Melle van der Meulen

A thesis presented for the degree of
Master of Science

Leiden Institute of Advanced Computer Science (LIACS)
&
Mathematical Institute (MI)
Leiden University

Supervisor I: Dr. Suzan Verberne
Supervisor II: Dr. Julian Karch
External supervisor: Ben van den Burgh

Acknowledgements

First of all, I would like to thank Suzan Verberne for being such a great supervisor during the thesis project. She have guided and helped me trough this journey. We had nice conversations, productive meetings and she triggered my enthusiasm for Natural Language Processing. It would be great to work with her again. Second, I would like to thank Julian Karch and Ben van den Burgh for supporting me with constructive feedback and their time for guiding me in this project. Furthermore, I want to thank Sarah Coombs. Sarah was of great importance of helping me with my academic writing and have given me great feedback. Finally, I want to thank my parents and my girlfriend for given me the support I needed trough thesis project.

Abstract

Product data can be useful to perform environmental impact assessments of product life-cycles. In order to automatize such assessments, this research is examining methodologies that encounter the challenges with the processing and classification of product data. We consider a large imbalanced and multilingual dataset with short and noisy product descriptions that have been labeled by human annotators. The product classes are hierarchically ordered and characterized by two levels. To treat the class imbalance we proposed two data enrichment methods on the training data. Oversampling and a web scraping method with a prior-filtering. The web scraper was parsing web data using a search engine and used Sentence-BERT with cosine-similarity to assess semantic relevant information. In addition, we proposed two classification methods, Support Vector Machines (SVM) and BERT. Both models were evaluated according to several experiments considering a flattened- and hierarchical classification approach of the products. In addition, we perform an extensive error analysis on the model results considering the SVM feature importance and the BERT attention weights.
The results showed that both models show similar flattened classification performance using the normal data, i.e. no data enrichment. SVM show better flattened classification performance after treated the class imbalance with data enrichment. BERT show poor performance using data enrichment and is overfitting the training data. Hierarchical classification improved the classification performance of BERT using oversampling. SVM did not benefit from the hierarchical classification approach and show better classification performance using flattened classification. As last, the error analysis have showed that the data consist of incorrect or subjective manual labeling. The SVM feature importance and BERT attention weights results suggest that non-representative tokens or out-of-vocabulary tokens have the tendency to decrease the classification performance.

Contents

1 Introduction

1.1 Motivation and problem definition

Transgressions of planetary boundaries have become central issues for policy-making. The growing population and wealth intensifies the environmental pressures and trigger the degradation of global ecosystems. The Intergovernmental Panel on Climate Change emphasising the degeneration of ecosystems and the need for structural change and action to reduce climate change (Pörtner et al., 2022). Scientific assessments have accelerated policymakers to submit legally agreements with targets to reduce climate change. These targets are putting increasing pressure on organizations to set ambitious goals in line with the policymakers and scientific assessments.

Metabolic is a consulting and research firm that is focusing on environmental development and tackling global sustainability challenges. They advise governments, businesses and NGOs on how to adapt to these environmental and strategic challenges (Metabolic, 2022). As part of Metabolic, the software department is working on a software solution to monitor environmental impacts associated with all stages of life cycles of a commercial product, process, or service (Ghoroghi, Rezgui, Petri, & Beach, 2022). In order to perform such impact assessments, Metabolic is required to process an organization's data containing product descriptions and map these with an Life Cycle Inventory database to assess their impacts (Wernet et al., 2016). The complexity for Metabolic lies in identifying the informative semantic features and classifying the product descriptions to perform impact assessments. The scope of their software solution is to automate this process with the use of machine learning methods.

To automatize such impact assessments, this research is examining methodologies that encounter the challenges with the processing and classification of product descriptions. We consider a labeled dataset with short and noisy product descriptions. The classes are imbalanced and hierarchically ordered. We propose two classification methods, Support Vector Machines and BERT, a deep learning model. The challenges cause by this data structure are: the identification of semantically informative features in short product descriptions, class imbalance and classifying hierarchically ordered labels. Consequently, we seek to answer the following research questions:

How can product data with hierarchically ordered and imbalanced classes be effectively classified?

This main research questions can be answered by examining the following subquestions:

1. How can we treat data imbalance to increase the quality of classifications?
2. How can we address the hierarchically oriented product classes?
3. How do deep learning methods differ from conventional learning methods for text classification?
4. What makes the learning methods effectively classify the product descriptions and what not?

1.2 Contributions

This research facilitates methods for the impact assessment of products and contributes to the automation of processing- and classification methods for imbalanced- and hierarchically oriented products descriptions. Additionally, we developed an API[1] which makes it possible to reproduce our methods and can be used for different product related classification tasks.

- All prior work appears to assesses different domains. This paper focuses specifically on the classification of product descriptions.
- We address the multilingualism in product descriptions.
- We propose a data enrichment method using publicly available data. We have developed a web scraper that uses web search results to enrich the imbalanced dataset. Additionally, we incorporate a quality measure that assesses the degree of similarity between the product descriptions and search results to increase the overall quality of data enrichment.
- We introduce a hierarchical classification framework to address imbalanced- and hierarchically ordered classes.
- Most researchers focused on the accuracy and correctness of classifications, while not addressing the imperfections. We evaluate the classification performances and we address the imperfections by analyzing the misclassifications considering an error analysis.

1.3 Structure of this paper

For the remainder of this paper, the report is structured into seven chapters: Background, Data, Methods, Results, Discussion, and Conclusion:

- Chapter 2. **Background and related work** provides the essential background information and key concepts to read through this report.
- Chapter 3. **Data** introduces and describes the dataset used for the experiments in the chapters that follow.
- Chapter 4. **Methods** describes the data preprocessing, data enrichment techniques and learning methods used for the experiments.
- Chapter 5. **Results** describes all experimental results.
- Chapter 6. **Discussion** evaluates the experiments and reflection on the error analysis.
- Chapter 7. **Conclusion** summarises all results and proposing future work for improvements.

[1] API: Application Programming Interface is a software tool and interface that facilitates the use of services between users and computers.

2 Background and related work

In this chapter we describe essential background information and key concepts required to read this paper. In addition, we outline related work concerning the challenges with short text data and text classification tasks with hierarchically ordered- or imbalanced class labels.

2.1 NLP paradigm

Natural Language Processing (NLP) refers to a domain of Artificial Intelligence that considers classification or extracting structural information from unstructured texts. NLP strives to give computers the ability to syntactic and semantic understanding of the complex and diverse nature of human language (IBM, 2020). The complexity lies in the challenging aspects of language such as multilingualism, the diverse forms of grammar or the ambiguity of words. NLP involves tasks such as Machine Translation, Named Entity Recognition, Text Classification and Sentiment Analysis, to name a few. In this research the main focus will consider text classification of product descriptions.

2.1.1 Word features

To process natural language, text data needs to be transformed into numerical representations in order to use machine learning methods. A simple and commonly used method is the Bag-of-Words (BOW) model. This method consider all the unique words in a collection of documents. Each document is represented as a vector with the number of dimensions equal to the number of unique words in the collection. Documents often contains a small part of the unique words present in the collection. As a result, most vectors consist of zero values and creates a sparse representation of the data. Sparse data increase the feature space and time complexity of models, resulting in inefficient and inaccurate computations (Jurafsky & Martin, 2021).

In addition, BOW representations can be used with methods that enhance the representations. Term frequency-inverse document frequency (TD-IDF) is a commonly used method for penalizing words that tend to be less relevant for the representation of documents. TF-IDF emphasizes word relevance by adding a weighting factor to words that appear more often in a document. This is counterbalanced by the number of documents in the corpus that contain these words (Qaiser & Ali, 2018; Jurafsky & Martin, 2021). The intuition is that frequently occurring words are often less relevant, e.g. stop-words. Furthermore, BOW representations can be enriched by using a combination of consecutive words to increase the feature space of the representations. N-grams conducts a sequence of n-consecutive words for creating a set of co-occurring words within a given window. This method can be useful for the presence of interrelated words, e.g. hard disk drive. In case of the latter, the BOW representation can be improved by incorporating 2- or 3-consecutive words as additional features, i.e. bigrams or trigrams.

BOW representations result in sparse vectors and ignores the syntactic and semantic relations between words. The distributional hypothesis emphasizes the role of context and suggest that words appearing in same contexts tend to have similar meaning (Huang, Hussain, Wang, & Zhang, 2019). Vector semantics utilize this principle by learning the representations of the meaning of words using word embeddings (Jurafsky & Martin, 2021).

2.1.2 Word embeddings

Word embeddings defines innovative methods that use self-supervision to map words into a continuous, dense- and low-dimensional vector space. A word embedding is a function that learns the context window of words to predict a target word, or vice versa. Using a simple neural network, the vectors are learned to represent the semantic relations between words to create low-dimensional vectors.

There are different techniques to construct word embeddings. We briefly discuss the most important methods. Commonly used methods, Word2Vec, GloVe and FastText are trained on a large text corpus with a fixed vocabulary and map each word to a fixed embedding (Mikolov, Chen, Corrado, & Dean, 2013; Pennington, Socher, & Manning, 2014; Bojanowski, Grave, Joulin, & Mikolov, 2016). Word2Vec consider two possible models, Continuous BOW (CBOW) and Skip-gram. The models use a simple neural network with different network architectures. CBOW trains a word vector by predicting a target word given its context, Skip-gram predicts the context given the target word (Mikolov et al., 2013). Instead of considering local properties, i.e. local context windows, GloVe encounter the ratio of global word-word co-occurrences probabilities to express the word relationship. The co-occurence matrix is factorized by minimizing a weighting function using least-squares. This results in weights corresponding to the word embedding (Pennington et al., 2014). Lastly, Word2Vec and GloVe fail to represent words that are out-of-vocabulary, i.e. rare words that have not been trained on. FastText uses a similar network architecture as Skip-gram, but encounters words as an n-gram of characters. Therefore, FastText is able to learn rare combinations of characters and words, making it possible to break down words into trained sub-words or characters to construct embeddings.

The discussed word embedding methods are commonly described as static word embeddings. Static word embeddings map semantic relations into a fixed embedding, e.g. apple-tree or grape-vine, but ignores word ambiguity and word order (Jurafsky & Martin, 2021). A commonly used example of word ambiguity is the following:

Open a **bank** account
On the river **bank**

This example emphasizes the limitations of static word embeddings with the occurrence of polsemy or ambiguous words (Devlin, 2021).

2.1.3 BERT

Recently, researchers in the field of deep learning have proposed innovative learning methods that attempt to solve the limitations experienced with static word embeddings and the improvement on modeling long-dependencies (Jurafsky & Martin, 2021). One of these methods is BERT, a deep transformer based model that learns bidirectional representations and long-dependencies from unlabeled text. Transformer models were originally proposed by Vaswani et al. in 2017. These models introduced an improvement on recurrent and convolutional architectures for addressing machine translation; translating text from one language to another. Two key innovations introduced trough transformers are the self-attention mechanism and positional encoding. Self-attention mechanism makes it possible to model word dependencies from an input sentence bidirectionally and simultaneously, thereby altering

the conventional recursively modeling from left-to-right. The attention function map word relevance with respect to other words in the sentence and creates an contextualized word embedding for each word. Additionally, the combination of multiple attention layers, i.e. multi-headed attention, makes it possible to learn the network different meanings of attention, e.g. grammar, vocabulary and conjugation. The encoder layer includes positional encoding, which makes it possible to learn positional differences of words. It adds additional positional encoding to the word embedding and allows the model to reason about the relative positions of any word. As a result, the model is able to capture different word senses: "work to live" and "live to work".

BERT consist of multiple encoders with 12 attention layers each. BERT-base has 12 encoders and BERT-large has 24 encoders, resulting in 144 and 288 attention heads, respectively (Devlin, Chang, Lee, & Toutanova, 2018). This illustrates the depth and complexity of each BERT model. Because of the computational complexity, BERT takes a maximum sentence length of 512 tokens as input and uses WordPiece tokenization to improve the handling of rare words (Wu et al., 2016). BERT operates in a transfer learning setting by pre-training a model on a large collection with self-supervision and subsequent fine-tuning on a particular task (Devlin et al., 2018). The original BERT authors proposed pre-training using a BookCorpus (800M words) and English Wikipedia (2500M words). Additionally, there are many variants that are pre-trained using multilingual or domain-specific data. At this moment, there are more than 2300 ready-to-use pre-trained BERT models available on Huggingface, a repository that offer such models (Wolf et al., 2019). This models can be used in a transfer learning setting and fine-tuned for downstream tasks, e.g. classification tasks.

2.2 Text classification

The goal of text classification is deciding what document can be assigned to an input or a fixed number of predefined labels. It involves the process of training a learning method, i.e. classifier f using labeled text data X and performing prediction tasks on a unseen dataset. A well known text classification problem is the detection of spam, which involves a binary classification task of assigning an email to one of the two classes: spam or not-spam (Jurafsky & Martin, 2021). In the case of multiple labels this task is considered as a multiclass classification task; classifying text data and supervising the learning method with $n > 2$ labels. This study focus on the latter and proposes classification methods for multiclass classification tasks. We will now discuss the employed methods.

2.2.1 Support Vector Machines

Support Vector Machines (SVM) is a non-probabilistic supervised learning method that uses a separating decision boundary. The optimal decision boundary corresponds to a line or hyperplane that separates training instances of either class, and is constructed using a margin that defines the largest distance between the hyperplane and training observations (Hastie, Tibshirani, & Friedman, 2009). The training observations that are closest between either class determines the margin and are often described as the supporting vectors. In practice, training observations are often not perfectly linear separable. Using a straight line or hyperplane may cause overfitting the training data, i.e. hard-margin method. To avoid overfitting, it can be advantageous to use a regularization term to penalize the classifier

and allow for some observations to be on the wrong side of the decision boundary, i.e. soft-margin (Hastie et al., 2009). This penalization exploit the trade-off between the margin width and the number of misclassifications to construct a better generalizing model.

To assess the best separating hyperplane and generalizing model, we optimize the hinge loss function with respect to the model parameters (eq.1). The hinge loss function is specifically useful for constrained optimization tasks such as SVM and seek to maximize the margins while allow for some misclassifications. This task can be described with the following optimization problem:

$$\underset{\beta \in \mathcal{R}^d}{\arg\min} \frac{1}{2}||\beta||^2 C + \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i) \tag{1}$$

where $||\beta||$ is the scaling parameter for the margins, C adds the regularizing term that allows for misclassifications, $\hat{y}_i$ is the decision boundary for class i and the hinge loss $\mathcal{L}$ corresponds to $\max(0, 1-\alpha)$ and N corresponds to the number of training samples.

Regularization is key to find the optimal trade-off between the model complexity and the ability to generalize with unseen data. This is often described as the bias-variance trade-off (Hastie et al., 2009). Two regularization methods are commonly used: L1- and L2-regularization. The intention of both methods is similar, regularizing less predictive features by shrinking the coefficients to zero or discarding them from the model. This decreases the model complexity by reducing the model variance in return for some bias to create a model that is less likely to overfit the training dataset. L1-regularization improves generalization by shrinking the sum of the absolute values of the model coefficients, $\sum_{j=1}^{p} |\beta|$. L2-regularization uses a penalty function based on the sum of squares of the model coefficients, $\sum_{j=1}^{p} \beta^2$. Following the properties of both constraints, L1 regularized coefficients are more likely to become zeros than L2 coefficients. As a result, the L1 penalty can lead to sparser models, whereas L2 penalty shrink the coefficients to become small and lead to non-sparse models.

One drawback to SVM with L1 regularization is the number of selected features p is bounded by the number of samples. Additionally, using this regularization method for sparse data, the L1 penalty seems to fail there is a strong correlation between features (Sharma, Verlekar, Ashary, & Zhiquan, 2017).

2.2.2 Fine-tuning BERT

The power of pre-trained BERT models is their ability to extract the rich word representations for downstream tasks. Fine-tuning BERT facilitates this by make it possible to use labeled data to train application specific parameters (Jurafsky & Martin, 2021). During fine-tuning, BERT captures all sentence information in a special embedding of [CLS] token that is specifically useful for sentence classification tasks. The final hidden state corresponding to this embedding is used as the aggregate sequence representation for classification tasks (Devlin et al., 2018). In addition, it is common to train a neural network by adding a linear classification layer on top of a pre-trained BERT model. This classification layer consist of a feedforward network with a Softmax layer to normalize the output nodes into a vector of

probabilities with size equal to n categories.

To find a set of parameters that maximizes the number of correct classifications, we optimize a loss function with respect to the model parameters. We use the categorical cross-entropy loss function (eq. 2). This loss function is stochastic and is specifically useful for this multiclass classification task. Cross-entropy quantify the differences between the probability distributions of class labels and makes a probabilistic decision. We use this loss function on a training set and minimize this function in order to improve the performance on the validation set.

$$\mathcal{L}(\hat{y}, y) = -\sum_{n=1}^{N}\sum_{c=1}^{C} y_n^c \log \hat{y}_n^c \tag{2}$$

Where C denotes the number of class labels, $\hat{y}_n^c$ denotes the predicted probabilities from the Softmax output, y_n^c is the truth class label and the total number of training samples are denoted by N.

To optimize this loss function with respect to the model parameters we propose Adam. Adam is an innovative Stochastic Gradient Descent method that optimizes model parameters individually by using adaptive learning rates. This innovation makes Adam highly effective for high-dimensional data tasks and deep learning models (Kingma & Ba, 2014). As a result, Adam is a commonly used optimization algorithm for text classification tasks and is recommended for fine-tuning BERT models (Devlin et al., 2018; Vaswani et al., 2017; Sun, Qiu, Xu, & Huang, 2019).

Finally, we address the bias-variance trade-off using dropout regularization. BERT consist of large network with many parameters that easily overfit with limited training data (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Dropout is a stochastic regularization method that randomly drops sets of nodes in the network and creates a smaller, simpler model to reduce overfitting. Additionally, dropout creates different network architectures and combines the optimal sets of nodes to construct a network with the lowest generalization error.

2.3 Hierarchical Classification

Most research in the field of data science is focused on classification tasks that consider a flattened structure of labels (Silla & Freitas, 2010). Flat classification assumes that the labels has no hierarchical structure and all classes are mutually independent. However, most product related data can be categorized to a mutual parent class. Ontologies such as the International Patent Classification system (IPC) (D'hondt, Verberne, Oostdijk, & Boves, 2017) or eCommerce databases are hierarchically ordered and consist of a pre-defined class hierarchy (Jahanshahi et al., 2021). This hierarchical structure is useful to structure large amount of data where instances can be assigned to multiple levels of categories. Conventional classification methods such as binary and multiclass classifiers cannot directly cope with the hierarchically ordered labels and often ignore the class hierarchy by predicting the classes by using a flat structure. The disadvantage to this is that these classifiers often need to discriminate between a large number of classes whenever a large hierarchical structure of labels is flattened. In order to address the class hierarchy, we proposed two approaches according to Silla and Freitas (2010):

1. Conventional multiclass classification and ignoring the class hierarchy by using flattened labels. This approach is used as a baseline.

2. Local classification or a top-down classification approach. We perform multiclass classification using local information. We train a classifier for the first-level or parent (most generic) class and training multiple classifiers at the second level or child (more specific) class. As a result, we can recursively predict the most generic class and uses this predicted class to narrow down the number of labels to be predicted for the child class. Because of the large set of labels, this approach is advantageous as it reduces the large set of labels into several small sets. Also, the classifier does not have to discriminate between a large set of labels.

2.4 Related work

2.4.1 Short text data

Challenges with short text data are expressed by Martinez-Gomez, Papachristoudis, Blauvelt, Rachlin, and Simhon (2021) using short product descriptions in the e-commerce domain. Most product names consist of abbreviations, non-homogeneous elements and multilingualism. More (2016) proposes sequence labeling models for detecting the features in product titles of e-commerce retailers. Noisy elements, ambiguous- or unrelated context words form major parts of each product title. For example, brand names may contain varying spellings of the same brand name or use abbreviations for a selection of products. Additionally, ambiguous words pose a key challenge for classifying product title.

Short text sentences are often characterized by a lack of contextual information. Chen, Hu, Liu, Xiao, and Jiang (2019) propose to integrate context using BOW and static word embedding representations for short sentences using a unified deep neural network model. This model incorporates additional contextual information from a Knowledge Base to enrich the semantic representations. Additionally, they propose two attention mechanisms that learns the importance of related concepts and quantify the similarities with the short text sentences. Finally, the authors demonstrate the effectiveness of integrating context information with different learning tasks for short text sentences.

In addition, Rued, Ciaramita, Mueller, and Schuetze (2011) enrich ambiguous words in news texts for Named Entity Recognition tasks with contextual information using a search engine. The authors emphasize the ability of search engines to make optimal use of the context information and use short text sentences as query to find relevant information. They demonstrated that search engines are effective for enriching ambiguous words to build robust Named Entity Recognition models in the domain of news data.

Oksanen, Cocarascu, and Toni (2021) describes the informative aspects in short product reviews for automatically extracting ontologies and meronomies using BERT. BERT is fine-tuned for aspect- and relation extraction tasks by processing product reviews from Amazon and utilizing domain-specific terms. The proposed model is evaluated by its annotating performance using a large movie recommendation dataset. The proposed method generalises well and outperforms the existing movie ontologies. Additionally, the authors emphasize the potential of meronomies to be specialised forms of ontologies.

2.4.2 Classification

One of the early text classification propositions discusses the linear separability of text categorization problems using BOW (Joachims, 1998). Text data is often high-dimensional and dichotomous which can be easily separated using a linear classifier according to Cover's theorem (Cover, 1965). The authors propose SVM with BOW and show that linear SVM are well suited for text classification problems.
In addition, Yu, Ho, Arunachalam, Somaiya, and Lin (2012) propose to use of SVM with BOW variations for classification using short product descriptions. The authors describes the limitations using conventional word features for the classification of product descriptions. In short text sentences, features such as stemming, stop-word removal or the use of weighting terms such as TF-IDF, may not be effective and affect classification performance. Contrarily, BOW with n-gram representations are predominantly effective for short product titles.

Linear classifiers appear to be effective for classifying product descriptions. However, product data often exhibit a hierarchically ordered data structure. In addition, patent data, medical records or web directories are typical examples of large hierarchical text repositories. These repositories consist of a large number of closely related classes that are often hierarchically ordered. Gao, Zhao, Ma, Tanvir, and Jin (2022) describe the challenging aspects for classification of such repositories and emphasize the effectiveness of integrating parent-child class relationships into classification models. The authors propose a LSTM-network that incorporates the semantic parent-child relationships as a joint embedding and use a multilayer perceptron with cross-entropy to train the network parameters. Their results demonstrated that incorporating the hierarchical class structure show better classification performance than using a flattened classification approach with, e.g. BERT.

The hierarchical nature of large text repositories often consist of long-tailed class distributions inherently. Class imbalance occurs when the majority of the class labels have only a few instances and are positioned in the tail of the class distribution. Feng, Fu, and Zheng (2018) illustrates the latter within the biochemical domain and describes the challenges with gene function prediction as a classification task. The authors consider the class imbalance and the hierarchical orientation of labels using a Gene Ontology. They propose a hierarchical multi-label classification method that encounter the hierarchical gene functions (class labels) as an local approach by training a multilayer perceptron classifier for each class independently with local information. Additionally, the class imbalance is treated using an oversampling method to increase the minority class instances for each training data set. As a result, the local classification approach outperform other hierarchical multi-label classification methods for classifying gene functions.

3 Data

The dataset consists of a collection of datasets from six mutually independent companies. These companies were selected to represent a variety of product classes, both food and non-food related. The data contains short product descriptions that are Dutch and English. The product descriptions were manually labelled by consultants from Metabolic. The labels corresponds to two predefined classes wich are included as two variables in the dataset. We processed the data by removing the missing values and incorrect product classes. In

addition, the data contains duplicate products. Companies can share a common product or valuable products can occur more often in the dataset. We emphasized the effect of identical products in the dataset, i.e. duplicates by using two datasets for our experiments. We used the true data set with duplicate products and a dataset without the duplicates.

3.1 Descriptive statistics

Table 1 show the dataset with the variables and descriptions. The dataset consist of 36, 165 product descriptions of which 24, 715 are unique. There are two product classes AnalyseCategorie and MAIA KeyProduct with 22 and 952 class labels, respectively. AnalyseCategorie class consider Dutch class labels, while MAIA KeyProduct class labels are English. Figure 1 show the distribution of tokens in the product descriptions. On average the commodities consist of 4.5 tokens per description.

Table 1
Description of the variables in the product dataset. Type indicates the data types for each variable in the dataset; integer (int) and string (str). The dataset consist of 36, 165 samples.

Variable	Type	Description
Bron	int	Describes the data source. The data consist of a collection of datasets from six mutually independent companies. The companies are anonymized using random company IDs, i.e. Bron.
Omschrijving	str	product descriptions; sentences with the semantic information that describes the commodities (24,715 unique product descriptions).
AnalyseCategorie	str	Product class with (n=22) labels in Dutch.
MAIA KeyProduct	str	Product class with (n=952) labels in English.
AC	int	Encoded AnalyseCategorie class
MAIA	int	Encoded MAIA KeyProduct class

3.2 Hierarchical structure of labels

The variable AnalyseCategorie, i.e. AC describes the generic product labels which corresponds to the first-level or parent class. Additionally, the variable MAIA KeyProduct, i.e. MAIA describes a large set of more specific product labels that are interrelated with the parent class labels. Figure 2 illustrates the parent- and child class relations for a random selection of labels and with the unprocessed product descriptions.

3.3 Data imbalance

Figure 3 shows the class distribution for both product classes. This figure illustrates the class imbalance and the class labels that can be considered as the majority classes, i.e. over-represented classes in the data or minority classes, i.e. under-represented classes in the data. The class imbalance is characterized by a long-tailed distribution of class labels. Class imbalance could affect the discriminating ability of the minority classes. In addition,

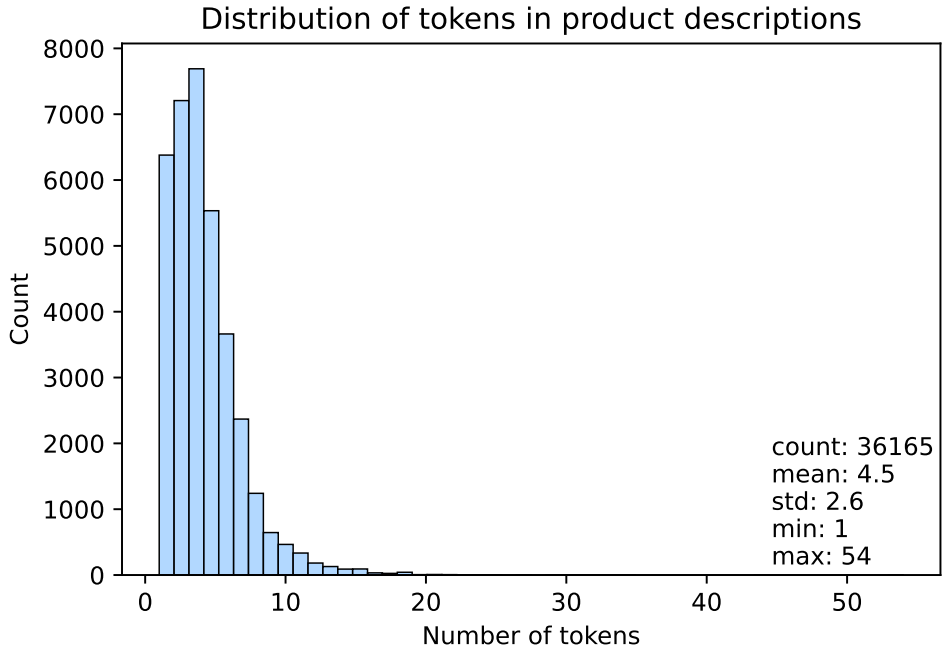

Figure 1. Distribution of tokens in the product descriptions including descriptive statistics. The mean number of tokens per product description of 4.5 emphasizes the small size of the product descriptions.

learning methods have the tendency to be biased towards the majority classes and often decrease the predicting performance (Weiss, 2004). Most learning methods have an accuracy oriented design, which are naturally more biased towards the majority classes (Fernández et al., 2018). Moreover, noisy data tends to have a greater effect on the minority classes, making it more difficult to discriminate between the classes (Weiss, 2004).

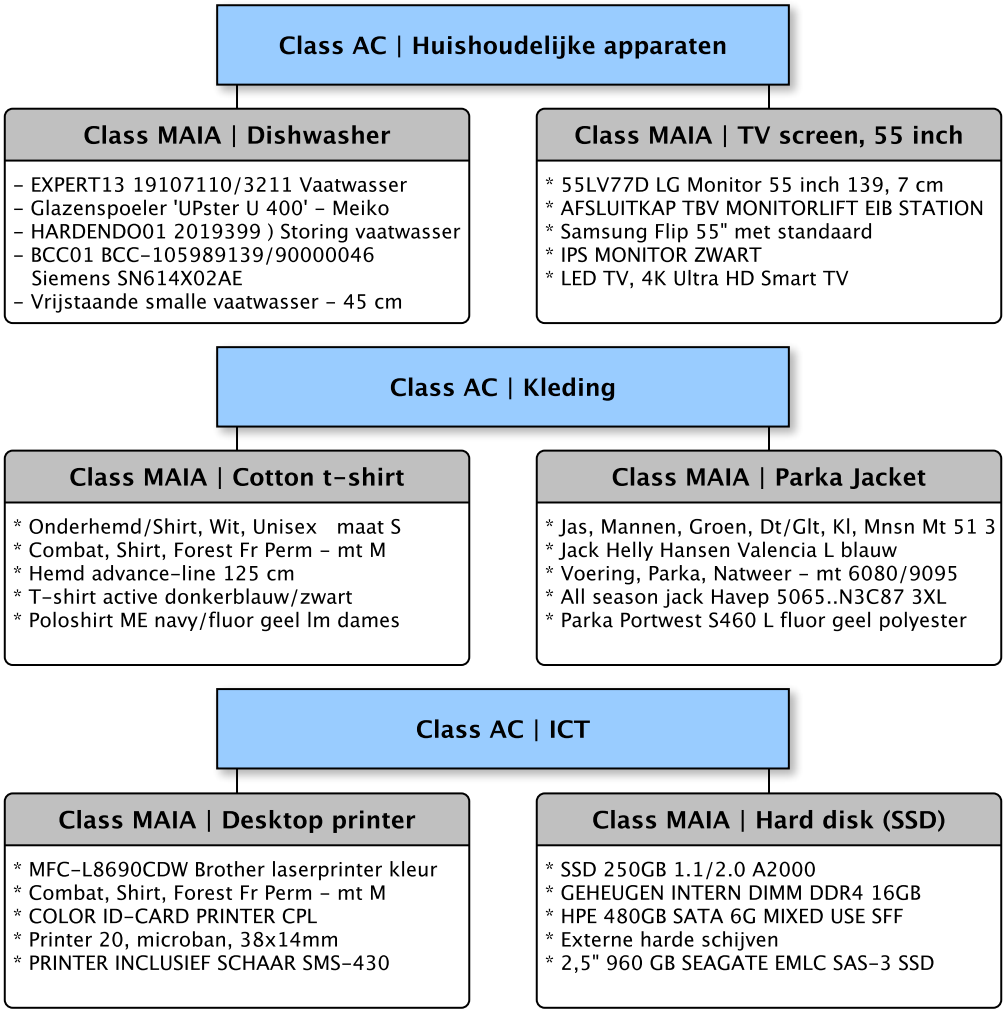

Figure 2. Tree structure to illustrate to hierarchical product classes. The top-level nodes (blue) describes the parent class AC for a selection of labels. The recursive leave nodes (gray) describes the related child class MAIA labels with a selection of unprocessed product descriptions.

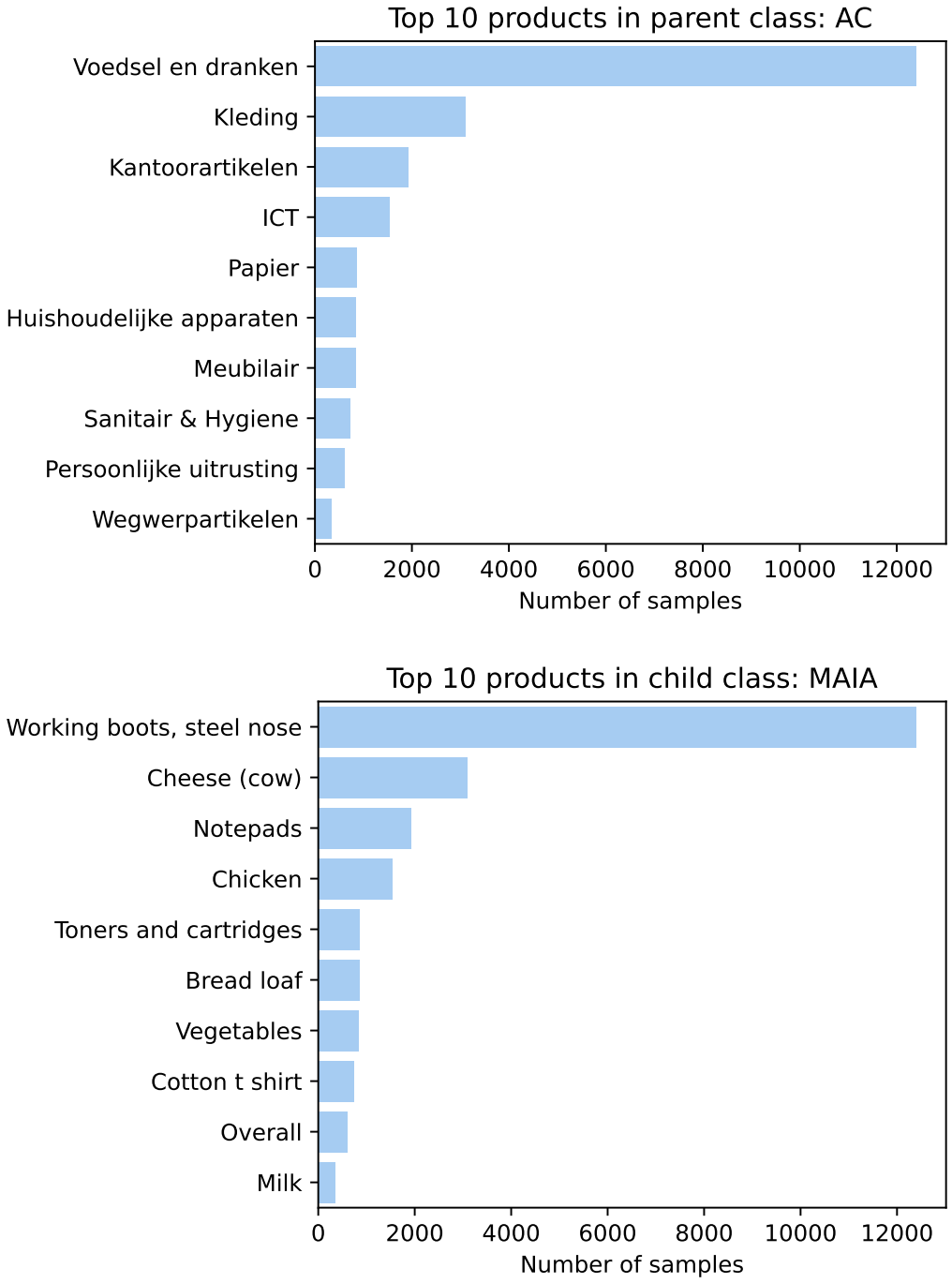

Figure 3. Bar plot with the 10 most frequent labels in the parent class AC and child class MAIA. The length of the bar represents the number of samples in each class label.

4 Methods

This chapter will describe the details of the proposed methods and experiments. We describe the data preprocessing, data enrichment, the classification methods and the experimental setup to illustrate the training aspects.

4.1 Data preprocessing

We preprocessed the data by removing all empty product descriptions, and all non-labeled products. Subsequently, preprocessing the product descriptions is advantageous to retain most informative features, while removing noisy elements that have no informative property. Table 2 shows the product descriptions before- and after data preprocessing.

The following preprocessing steps have been taken:

- Case folding: mapping all characters to lower case. The product descriptions show capitalized sentences or a combination of lower- and uppercase words and characters. Due to this unstructured use of capitalization it provides little to no valuable information.
- Remove punctuation.
- Remove special characters.
- Remove tokens with length $n = 1$.
- Remove tokens containing only digits.

In addition, we examined options to normalize words by using a standard root form; lemmatization or stemming (Jurafsky & Martin, 2021). However, in case of small text descriptions, using normalization may result in incorrect semantic forms. Yu et al. (2012) have shown that normalization of small (products) descriptions results in disambiguation problems. For example, using stemming, products containing the word: "recorder" from the category consumer electronics change into "record" which corresponds to the product category music. Because product descriptions are short, normalization seems to remove essential information that make some words less discriminating for the related category.

4.2 Data enrichment

The data imbalance was treated according to a data level approach (Fernández et al., 2018). This approach aims to rebalance the class distribution or is incorporating new data. Both results are useful treat treating skewed label distributions. Therefore, two methods were proposed. Oversampling, and a data enrichment method. This method used web scraping to collect new product descriptions from existing ones using information from the world-wide-web.

Table 2
Sample of the dataset illustrating the product descriptions before- and after preprocessing the data. AC and MAIA show the encoded product classes.

Description	AC	MAIA
KRUIDEN BASILICUM	18	53
Smitvis haring+uitjes 5 st msc	18	406
Snelheidsmeter V-MAXX - inclusief adapter en statief	4	801
110389 - Lipton feel good select.bosvruch.ht 25s	18	422
Broek, Waterproof, Olive-Drab, Mt m	7	430
kruiden basilicum	18	53
smitvis haring uitjes st msc	18	406
snelheidsmeter maxx inclusief adapter en statief	4	801
lipton feel good select bosvruch ht 25s	18	422
broek waterproof olive drab mt	7	430

- Encoded AC categories: 4, 7 and 18. These encoded labels correspond to: ICT, Kleding and Voedsel en Dranken, respectively.
- Encoded MAIA categories: 53, 406, 422, 430, 801. These encoded labels correspond to: Basil, Herring, Icetea, Jeans JRC BOP, and Speaker stand large (6m height), respectively.

4.2.1 Oversampling

Oversampling increases the number of samples in the the minority classes and resamples the data to create a more balanced distribution of class labels. We used oversampling on the training set after splitting the data into a train- and test set. Oversampling on the complete dataset would result in identical observations to be present in both the train and test sets. This allows learning methods to simply memorize observations and result in a poor ability to generalize. We performed oversampling on class labels when the size $\leq$ average class size. Oversampling was proposed for both parent- and child classes with a resampling size of 10% of the maximum class label size.

Oversampling makes it possible to resample existing observations, but is not able to enrich the data with new information. Moreover, resampling existing observations can enhance overfitting problems since learning methods may focus too heavily on oversampled class labels (Fernández et al., 2018; Chawla, Bowyer, Hall, & Kegelmeyer, 2002). Therefore, we proposed an additional data enrichment method that incorporates new informative samples using Google search results and makes it possible to enrich our data with public available data.

4.2.2 Web scraping

To enrich the training data we proposed a web scraping method that exhibits semantic related product data using a search engine and process the data into qualitative new features. We used the Google search API to utilize our web scraper and parse HTML data from the headings in the first page with `BeautifulSoup` (Richardson, 2007). We proposed web

scraping when class label size $<$ average class size.

As discussed in Section 2.4, web search engines exhibit large margins of error when insufficient context is provided. Therefore, we enriched the queries using a combination of product descriptions and related child class labels. The child class labels consists of detailed product information and provide additional context. We combined each combination to a single sequence and used it as input query for the search engine.

Furthermore, search engines like Google, have the drawback they are limited to a fixed amount of rate requests in a given amount of time before a CAPTCHA[2] protocol intervenes in the search operation. As a result, it was not possible to automatically perform a large number of search requests and it was not possible to enrich the complete set of categories when this sample size exceeds the rate request limit. To comply with this rate request protocol, we used a time-out moment after every search operation to simulate more human-like search behavior. The time-out was randomly chosen using a uniform distribution of values between 1 and 5 seconds.

In addition, search results are at times noisy with irrelevant results that need to be addressed using a post-filtering method. We used sentence-BERT with cosine similarity to measure the semantic similarity between the product descriptions and the processed web results. Sentence-BERT is a modification of BERT that uses a different network structure to derive semantically relevant sentence embeddings that can be compared using cosine-similarity (Reimers & Gurevych, 2019). We evaluated to what degree the search results are similar to the product descriptions using this similarity score and set the semantic similarity threshold at $\epsilon \geq 0.8$ following (Jinfeng, Shouling, Tianyu, Bo, & Ting, 2019)

4.3 Classification

All non-hierarchical classifiers addressed the classifications tasks as a flattened multiclass classification task and ignored the hierarchical structure. The classifiers were trained on both product classes to construct two distinctive classifiers for each of the classes.

4.3.1 Support Vector Machines

We used SVM as a baseline to compare the classification experiments. We used linear SVM with BOW including the option for Bigrams and TF-IDF. In addition, we experimented with L1- and L2-regularization using a wide range of parameter values. We use Stochastic Gradient Descent for optimizing the hinge loss function and we used a one-versus-rest training method to address the multiple classes. The reasoning for the one-versus-rest method enables multiclass data into one binary dataset for each class and train each classifier as a binary task (Bishop, 2006). We performed all our experiments using `Scikit-learn` (Pedregosa et al., 2011).

[2]Security measure that prevent automated programs from abusing online services (Ahn, Blum, Hopper, & Langford, 2003).

Table 3
Overview of proposed BERT models with the corpora that have been used for pre-training.

Model	Corpus
bert-base-uncased	BookCorpus, Wikipedia (EN)
bert-multilingual-uncased	Wikipedia (102 languages)
bertje	5 different corpora (Dutch)*
distilbert-base	BookCorpus, Wikipedia (EN)

* These corpora include: Collection of contemporary and historical fiction novels, a Dutch news corpus, a multi-genre reference corpus, articles of 4 Dutch news websites (2015-2019) and Dutch Wikipedia (2019).

4.3.2 BERT

To fine-tune BERT with our data, we selected pre-trained models from the Huggingface repository. In order to experiment with our multilingual product data, we used an English and multilingual model (Devlin et al., 2018), a Dutch model (de Vries et al., 2019) and a distilled version of BERT to study the effect of a smaller BERT model on the fine-tuning performance (Sanh, Debut, Chaumond, & Wolf, 2019). All models with corresponding pre-training corpera are shown in Table 3.

The product descriptions were tokenized using the pre-trained BERT models with the maximum length set to 64. WordPiece tokenization was used for out-of-vocabulary tokens by splitting words into subwords or characters to match the input tokens with the BERT vocabulary (Wu et al., 2016). The sub-words or characters were added using the identifying prefix: "##", e.g. "playing" can be split into "play" and "ing".

Lastly, the classification layer take the embedding of the [CLS] token from the last encoding layer as input and transforms into a vector with the size of the number of class labels prior to applying Softmax (Devlin et al., 2018).

Parameter settings

We optimized the training parameters by experimenting with different learning rates for the Adam optimizer. The learning rate is often the single most important hyperparameter to increase the model performance, since it controls the weights adjustments of the network and enhances convergence towards an optimal set of parameters (Bengio, 2012). Over a large number of epochs[3], a small learning rate may overfit the data, while a large learning rate may underfit the data (Smith, 2018). We used a learning rate warmup over the first 2 epochs and a linear decay of the learning rate according to Devlin et al. (2018). In addition, we used early stopping to regularize the loss function whenever the validation loss tended to increase and model start overfitting the data. We used the principle of early stopping after 5 consecutive epochs of increasing validation loss. We set the number of epochs to 30 as default.

[3]One cycle through the full training set to train the internal parameters of the network.

Second, we experimented with the dropout parameters within the BERT model architecture. We performed dropout regularization on both the hidden- and the classification layers.

Lastly, we experimented with different batch sizes. Batch size defines the number of samples to be propagated through the network in each batch and is related to the total number iterations per epoch. Batch size is sensitive to class imbalance. Small batch size may cause bias towards the majority class, while a size that is too large result in poorer generalization (Keskar, Mudigere, Nocedal, Smelyanskiy, & Tang, 2016). Moreover, batch size must also be adapted for hierarchical classification. The parent class labels are imbalanced and minority classes show a small number of training samples. Because the batch size cannot exceed the number of training samples, it must be adjusted for the training size. We used a batch size that corresponded to a minimum of 30 iterations per epoch and a maximum batch size of 128. Using this setting, the minority classes have an suitable number of iterations per epoch and the majority classes can be trained more computational efficient using a large batch size of 128.

The experiments were configured and conducted using `Tensorflow` and `Keras` (Abadi et al., 2015). Each were run on a TPU machine to increase modeling efficiency and computation time (Jouppi et al., 2017).

4.3.3 Hierarchical classification

To address the hierarchical class structure we trained the classifiers in a top-down matter as discussed in Section 2.3. We recursively trained each classifier using the predictions from the parent class to narrow down the number of labels to be predicted for the child class in a top-down matter. Therefore, we first trained a generic classifier using the parent class as root model. In addition, we train classifiers for each of the parent class labels (n=22) and conditioned the model on the related subset of child class labels (eq. 3).

$$f(X_i) = y_{(child|i)} \tag{3}$$

Where X is the training data that corresponds to parent class label i. The response variable y consist of all labels in the child class conditioned on parent class label i.

4.4 Experimental setup

The experimental setup consisted of the following parts:

- Data and hyperparameters
- Baseline classifications
- Hierarchical classifications
- Data enriched classifications
- Generalization on LOO samples between clients

4.4.1 Data and hyperparameters

The dataset corresponded to the collection of datasets from different companies. Because of this the data is naturally sorted by company and consisted of several rows with mutually dependent products. If we train the data in batches, e.g. classification with BERT, some batches will not be representative for the overall dataset since it may contain only products related to some company. Therefore, we shuffled the data to reduce the variance and to decrease the chance of overfitting the data.

Furthermore, we split our dataset into a training and test set using a split proportion of 90% and 10%. Splitting the data makes it possible to train and optimize the models using most of the data, while evaluating the models on a unseen test set. Training the BERT models uses an additional validation set to provide an unbiased evaluation of training the model while optimizing the model hyperparameters. Therefore, we split the train set to create an additional validation set. We used 80% of the data for training, 10% for validation, and 10% for testing.

Because we used the true dataset that containing identical products, the test data may include some products that are also present in the training data. Therefore, we evaluated model performance for the baseline SVM classification model on data with- and without the occurrences of duplicate products. Using this evaluation we were able to emphasize the difference in classification performance between a realistic data set including duplicate products and a modified dataset that excludes all duplicates. We expected that including duplicates would result in the overestimations of the model performance, as the models simply recognize products in the test set that have been trained on. For the rest of the experiments we used the true data set that consist of the true product data which include duplicate products.

Support Vector Machines The optimal hyperparameters for Support Vector Machines classifiers were determined using GridSearchCV with 5-fold cross-validation from the `scikit-learn` library. Table 4 shows the hyperparameters to be optimized and the corresponding search grid.

Table 4
Hyperparameters grid for the SVM classifier and grid search range. Word features describes the selection of BOW representations. The regularization terms relate to the L1- and L2-penalty. Penalty strength describes the degree of regularization using a range of values, where a larger value specify stronger regularization.

Word features	Penalty	Penalty strength (C)
CountVectorizer, Bigram, TF-IDF	L1, L2	(9e-9, 9e-4)

BERT The optimal hyperparameter settings for fine-tuning BERT are estimated using a grid search on predefined set of hyperparameter values. Table 5 show the hyperparameters to be optimized and the corresponding search grid.

Table 5
Hyperparameters grid for the BERT classifier and grid search range. Model describes the pre-trained BERT models that have been utilized. DO describes the dropout regularization with a range of values that indicates the proportion of nodes that is omitted during training. Optimizer corresponds to the proposed optimization algorithm Adam and LR described the range of learning rate values.

Model	DO	BS	Optimizer	LR
bert-base, bert-multilingual bertje, distilbert-base	0, 0.1, 0.2, 0.3	8, 16, 32, 64, 128	Adam	(5e-5,1e-5)

4.4.2 Baseline classifications

The baseline classification corresponds to the classification models using the flattened labels of the separate product classes. This results in two different classifiers, both supervised using only one of the classes and the corresponding labels. This baseline classification is required for examining the overall performance of each model, and the relation to each of the class related categories. Consequently, these results give a deeper insight in the effect of the class size, and the model performance.

4.4.3 Hierarchical classifications

We used hierarchical classification as a top-down approach according to Section 4.4.3.

4.4.4 Data enriched classifications

All models were compared for differences in model performance before- and after the data enrichment on the train set. The test set was untouched to enhance unbiased results and evaluation of the classification performance between the proposed models.

4.4.5 Generalization on LOO samples

Taking the leave-one-out samples into consideration, we trained a model using a subset of companies and tested model performance using an unseen data set corresponding to a held-out company during training. By consequently leaving distinctive companies out of training, we examined the model generalizing ability for new data sets and unseen commodities that are company specific.

4.4.6 Evaluation metrics

In order to address the classification performance we used a selection of classification metrics that encounter the class imbalance. As mentioned in 3.3, classification accuracy place more weight on the majority classes than on the minority classes, which makes it difficult to have an unbiased, and objective view on the model performance. Moreover, accuracy does not account for any False Positives (FP) or False Negative (FN) predictions, which are equivalent to, respectively, Type-I and Type-II errors in hypothesis testing. Therefore, it is advantageous to measure the classification performance using a more strategic set of scoring

metrics. We used the following scoring metrics to evaluate our classifiers and construct a more robust evaluation of our classification performances:

Precision Proportion of assigned labels that are correct; True Positives (TP) to all the predicted positive (TP + FP).

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{4}$$

Recall Proportion of the relevant labels that were assigned; True Positives (TP) to all positives in the data (TP + FN).

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{5}$$

F1-score Harmonic mean of Precision and Recall, which give a balanced score for the ratio of True Positives (TP) to all predicted- and true positives in the data (TP + FP + FN).

$$F1_i = \frac{2 \cdot TP_i}{TP_i + FP_i + FN_i} \tag{6}$$

Evaluation hierarchical classification To evaluate our hierarchical classification, we used a modified version of the precision, recall and f1-scores that incorporates the predictions of all hierarchical classifiers in a top-down matter. First, we treat the root model R as a flattened classifier. The performance was assessed using the conventional evaluation metrics in equation 4, 5 6. Second, to evaluate a local classifiers L_n we used a modified version from Kiritchenko and Famili (2005) to incorporate all classifications per parent class label.

$$P_{L_i} = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{P}_i|} \tag{7}$$

$$R_{L_i} = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{T}_i|} \tag{8}$$

$$F1_{L_i} = \frac{2 \cdot L_i P \cdot L_i R}{L_i P + L_i R} \tag{9}$$

As last, we assess the hierarchical performance per parent class label i by addition of the root- and local results and using the average score. Consequently, we are able to evaluate the top-down approach to assess hierarchical classification; combining the predictions at root level R and narrow down using the predictions at lower level L_n.

$$hP_i = \frac{P_R + P_{L_i}}{2} \tag{10}$$

$$hR_i = \frac{R_R + R_{L_i}}{2} \tag{11}$$

$$hF1_i = \frac{F1_R + F1_{L_i}}{2} \tag{12}$$

Macro scores Because the product classes were imbalanced, scores tended to be more biased towards the majority classes. This was exhibit using Macro-averaging. Each class contributed equally to the performance score and created an unbiased view on the general model performance. Consequently, we used macro scores as default for all experiments if not stated otherwise.

Weighted scores Weighted-average scores emphasizes the class size and consider an average score with respect to the actual occurrences of the class labels in the dataset. Using weighted-averaging, we emphasized the class support and put more weight on the majority class labels.

Deterministic- and stochastic methods The proposed classification models use different loss functions. SVM is deterministic, while BERT is stochastic and consists of random processes. Therefore, we used 5 repeated runs for the BERT models and used average scores with confident intervals to construct unbiased estimates on classification performance.

5 Results

In this chapter we will discuss our findings from the data enrichment and classification experiments. First, the data enrichment results are discussed and evaluated using the classification models for both flattened- and hierarchical classification. Secondly, the best performing combination between the data enrichment and classification models are compared using their generalizing ability using a LOO set. Lastly, we conduct an error analysis on the best performing models and examine their limitations according to the number of misclassifications and feature importance or attention assessment.

5.1 Data Enrichment

5.1.1 Web scraping

To demonstrate the web scraping results Table 6 show a selection of product descriptions that satisfy the condition for web scraping.

Table 6
Sample of products that are used as input for data enrichment using web scraping method.

Omschrijving	AnalyseCategorie	MAIA KeyProduct
1. zwarte kip advocaat	Voedsel en dranken	Advocaat
2. eliterna boerenjongens op brandewijn	Voedsel en dranken	Advocaat
3. aperol	Voedsel en dranken	Advocaat
4. de kuyper tokkelroom original	Voedsel en dranken	Advocaat

This sample illustrates the challenge of collecting qualitative data using web scraping. First, the term “Advocaat” is polysemous; in this example it refer to a beverage, but without context it could refer to a lawyer. Second, brand names are ambiguous, e.g. “zwarte kip” could refer to chicken while in this example it is a beverage brand. Third, the sample

contained incorrect labels, “boerenjongens op brandewijn” and “aperol” are beverages but not related to Advocaat. To account for these semantic challenges, we utilized the proposed methods in Section 4.2.2. Table 7 show the web scraping results for the data sample and the following queries:

1. Query: “zwarte kip Advocaat”
2. Query: “eliterna boerenjongens op brandewijn Advocaat”
3. Query: “aperol Advocaat”
4. Query: “de kuyper tokkelroom original Advocaat”

Table 7
Selection of processed web scraping results with the sentence-BERT cosine similarity scores. Results with scores > 0.8 are highlighted and satisfy the condition for data enrichment. The index numbers correspond to the queries.

Results	CS
1. **zwarte kip advocaat cl dirckiii**	**0.911**
1. zwarte kip advocaat 50cl voordelig kopen drankdozijn nl	0.685
1. richard korver advocaten	0.745
2. **achtigen vruchten op brandewijn boerenjongens eliterna**	**0.959**
2. eliterna boerenjongens youtube	0.709
2. schwartz advocaten	0.730
3. **aperol bidfood**	**0.883**
3. aperol spritz simone kitchen	0.529
4. **de kuyper tokkelroom fles cl bidfood**	**0.881**
4. de kuyper gedistilleerd mitra drankenspeciaalzaken	0.657
4. advocatenkantoor de baarsjes	0.472

The results show two scraped sequences per query. The cosine similarity scores quantify the semantic similarity with the product descriptions using sentence-BERT embeddings. We see notable differences according to the cosine similarity scores. Web results corresponding to a cosine similarity score > 0.8 show semantic related descriptions. We observe that Sentence-BERT with cosine similarity threshold enhances the quality and is discriminating polysemy and ambiguous words, e.g. Advocaat in context of law is excluded from the data set. Furthermore, the web scraper enriches the product descriptions with new semantic similar variations for both correct- and incorrect class labels.

5.1.2 Descriptives

Table 8 show the data enrichment effects on the train set. We observe that the web scraping results do not differ between the parent class and the child class. As mentioned in 4.2.2 the web scraping utilizing the imbalance within the child class labels and enrich the data by using the deepest level of class labels. Furthermore, we observe that the combined data enrichment does not result in the largest data set. The oversampling is used after the data has been enriched using web scraping results, which resulted in less imbalanced class labels

to be oversampled. Lastly, we observe a large difference between the training size of both product classes using oversampling. The child class corresponds to a training size of almost three times larger than the parent class. This difference suggest that the child class consisted more imbalanced class labels and resulting in heavier oversampling of products.

Table 8
Effect of data enrichment on train set for both classes using the true dataset. Comparison of the enrichment methods: Normal, Oversampling, Webscraping and a combination of both, respectively. The results are describing the number of training samples after data enrichtment for both product classes.

	Train size			
Class	Normal	Oversampling	Web scraping	Combination
AC	32,548	63,328	40,880	78,284
MAIA	32,548	155,326	40,880	153,941

5.2 Classification results

5.2.1 Flattened classification

Support Vector Machines The first objective was to construct a baseline SVM model that was trained and optimized using a range of hyperparameter settings. We first illustrate the classification results for the models using default hyperparameter settings. Table 9 shows the classification results using default settings and the use of data with- and without identical products, i.e. duplicates. According classification metrics, the parent class model show significant better performance with respect to the child class model. First, considering the parent class model without duplicates we see that the precision score outperforms the recall score. This high precision score indicates that the false positive rate is low and the model is able to correctly classify the assigned labels. However, the relatively poor recall score indicates that there are many false negatives and suggest the model is unable to correctly classify the total number of assigned labels. Second, considering the child class, the results show poor scores on all classification metrics. The child class is characterized by its large set of imbalanced class labels and suffer from insufficient training samples.

In addition, the results show that including duplicates improves overall classification performance principally because of the increasing recall scores. The improvement of the recall score indicates that including duplicates decreases the number of false negatives and enhance the ability to correctly classify the assigned labels. If we compare the models with respect to data without duplicates, we observe that including duplicates result in a better performing SVM model by approximately 10% and 15% of the parent class and child class, respectively. However, including duplicates in the data may cause overestimation of the models performances since the test set may contain similar observations.

Table 10 illustrates the results of the SVM models with optimized hyperparameters. The optimization results disclose an optimal word representation for the parent class using BOW with TF-IDF and suggest that weighting the terms increase generalizing performance

Table 9
Performance SVM classification on test set using default hyperparameter settings and BOW. The results show the performance of both classification models for the parent- and child class. We emphasize the effect of identical products by training and testing the models on data with- and without duplicates. The scores consider the macro-averaging scores.

Data **without** duplicates				Data **with** duplicates			
Class	Precision_M	Recall_M	F1_M	Class	Precision_M	Recall_M	F1_M
AC	0.901	0.665	0.730	AC	0.897	0.739	0.795
MAIA	0.599	0.590	0.573	MAIA	0.639	0.648	0.626

according to the classification metrics. The child class model appears to improve when bigrams are used as additional feature. Furthermore, the results show that L2-regularization increase the overall classification performance for both models using data with- and without duplicates. We observe that the models using data without duplicates are stronger regularized. We suggest that this data set contains more imbalanced class labels which result in stronger regularization to decrease the number of misclassification on the minority classes (He & Ma, 2013). In addition, the increasing recall scores for all models suggest that regularization increase the overall model performance for imbalanced data by decreasing the number of false positives and improve correct classifications for the minority classes.

Table 10
Performance SVM classification on test set using optimized hyperparameter settings and best word representation. The results show the performance of both classification models for the parent- and child class according to the data with- and without duplicates. The scores consider the macro-averaging scores.

Data **without** duplicates

Class	Feature	P	C	Precision_M	Recall_M	F1_M
AC	BOW+TF-IDF	L2	1e-5	0.922	0.774	0.822
MAIA	BOW+Bi+TF-IDF	L2	1e-5	0.602	0.603	0.580

Data **with** duplicates

Class	Feature	P	C	Precision_M	Recall_M	F1_M
AC	BOW+TF-IDF	L2	9e-6	0.912	0.840	0.868
MAIA	BOW+Bi+TF-IDF	L2	9e-6	0.699	0.691	0.679

*Bi represents Bigrams.

BERT The first objective was to find the optimal pre-trained BERT model from the Huggingface repository. The models were selected by relevance, i.e. language diversity and were fine-tuned and optimized using the hyperparameter settings in Table 5. For convenience, the models were trained using the parent class according to the dataset with duplicates, i.e. the true dataset, and were compared by classification performance using a macro-averaging

F1-score. The results of the best performing classification models are showed in Table 11. Bert-base-multilingual-uncased show the highest proportion of corresponding tokens with 18.2% which corresponds to the best classification performance on the test set. The results suggest that most of the product data vocabulary is unknown for the pre-trained BERT models and will produce embeddings based on sub-tokens or characters. Moreover, we observe that most F1-scores are quite similar which indicates that there is no significant difference using BERT models that are pre-trained using English, Dutch or multilingual text data.

Table 11
Classification performance of multiple pre-trained BERT models on the test set for parent class AC using data with duplicates. Data_{vocab} describes the proportion of tokens in the dataset matching with the pre-trained BERT vocabularies. F1_M describes the macro F1-scores on the test set using the best model results from Appendix A. $t(s)$ gives an indication of the average training time in t seconds.

Model	Data_{vocab}	F1_M	t(s)
bert-base-uncased	14.5%	0.829±0.015	471±39
bert-base-multilingual-uncased	18.2%	0.832±0.030	460±27
bertje	11.0%	0.830±0.028	473±26
distilbert-uncased	14.5%	0.815±0.028	493±14

We experimented with the model architecture by utilizing dropout regularization and the learning rate of the Adam optimizer to assess the best possible classification model. Table 12 illustrates the results for the optimized classification models for both classes utilizing the data with- and without duplicates. Comparing the multilingual BERT model in Table 11 with the optimized model using the data with duplicates, we found that the performance on the F1-score increased with approximately 4%. These results suggest that optimizing the hyperparameters does not result in a significantly better classification model. Optimization of the parameters for a BERT model increases the training time substantially. It is questionable if optimization of these parameters give the desirable results.

Furthermore, a comparison of both class models results in BERT tending to have stronger dropout regularization for the parent class, i.e. 30% random dropouts in the linear layer. This regularization strength suggests that the parent class model has a tendency to overfit the data using a small set of imbalanced labels with many samples. In addition, the lower optimal learning rates emphasizes the model sensitivity for overfitting the data. Contrarily, the child class models are less regularized and show higher optimal learning rates. The results indicates that dropout regularization is not effective for imbalanced data containing a large number of minority classes.

The model performance between the two datasets indicates that including duplicates in the dataset increase the number of correct classifications. More concretely, the parent class model show a significant increase in recall score with an approximate increase of 22% in relation to the data without duplicates. This suggests that by including duplicates, the model is able to predict more labels correctly according to the drop in false negatives. This high recall score contributes to the overall performance, i.e. F1-score. In addition, the child

class model show a moderate increase in F1-score according to approximately 8%.

Table 12
Peformance BERT classification on test set using optimized hyperparameter settings. We use a multilingual pre-trained BERT model for the fine-tuning using the data with- and without duplicates. The results show the classification performance of both classification models for the parent- and child class labels.

Data **without** duplicates

Class	BS	LR	D_h	D_l	Precision_M	Recall_M	F1_M
AC	128	4e-5	0.1	0.3	0.817±0.013	0.707±0.040	0.735±0.033
MAIA	128	5e-5	0.1	0.1	0.649±0.007	0.643±0.018	0.628±0.011

Data **with** duplicates

Class	BS	LR	D_h	D_l	Precision_M	Recall_M	F1_M
AC	128	4e-5	0.1	0.3	0.887±0.009	0.860±0.013	0.868±0.006
MAIA	128	5e-5	0.1	0.1	0.699±0.007	0.695±0.006	0.682±0.007

5.2.2 Comparing data enrichment methods

The flattened classification experiments show the differences between SVM- and BERT classification models. We experimented with various datasets using data enrichment. Table 13 shows the results of the optimized SVM and BERT models utilizing the proposed data enriched methods on the training set. The results are discussed per data enrichment method using the normal data as baseline:

- **Normal:** The SVM and BERT models show similar classification performance on the test set for classifying the class labels according to the F1-scores.
- **Oversampling:** Oversampling does not appear to be effective for most models. Oversampling affect the classification performance of both models on the parent class, i.e. data with relatively less imbalanced classes. Moreover, we observe that oversampling has a considerable negative effect on the classification performance of the BERT models for both classes on the test set. The results suggest that oversampling increases the likelihood of overfitting in the training data for the parent class, but it slightly improve the SVM model performance considering the child class.
- **Web scraping:** Enriching the training set with web results increases the classification performance of the SVM models for both the parent- and child class models on the test set. The BERT classification results suggest that both models are overfitting the training data by incorporating new product data and increase the number of misclassifications for both classes.
- **Combination:** The combination of oversampling and web scraping show good results for the SVM child class model. We observe that this enrichment combination has increased the recall score by approximately 16% with respect to the same model using

the normal data. Contrarily, BERT models show again poor results and suggest that both BERT models are overfitting the training data using a combination of data enrichment methods.

By observing all flattened classification results, we found that the BERT models have the tendency for overfitting the training data after data enrichment and show poor generalizing performance on the test set. The SVM models show overall good classification results on the test data. Applying a combination of web scraping and oversampling to treat the class imbalance appear to be most effective for SVM child class model resulted in a significantly better classification performance on the test set.

Table 13
Flattened classification results for the optimized SVM- and BERT classifiers on the test set. The classification models were trained using training data that have been treated for class imbalance with the proposed data enrichment methods. The performance is described using the macro scores.

Model	Type	Class	Precision_M	Recall_M	F1_M
SVM	Normal	AC	0.912	0.840	0.868
SVM	Oversampling	AC	0.893	0.843	0.862
SVM	Web scraping	AC	0.939	0.869	0.895
SVM	Combination	AC	0.875	0.883	0.871
BERT	Normal	AC	0.887±0.009	0.860±0.013	0.868±0.006
BERT	Oversampling	AC	0.881±0.014	0.770±0.007	0.805±0.008
BERT	Web scraping	AC	0.896±0.013	0.829±0.001	0.848±0.006
BERT	Combination	AC	0.892±0.008	0.841±0.024	0.860±0.017
SVM	Normal	MAIA	0.694	0.692	0.676
SVM	Oversampling	MAIA	0.708	0.695	0.686
SVM	Web scraping	MAIA	0.795	0.786	0.777
SVM	Combination	MAIA	0.811	0.803	0.792
BERT	Normal	MAIA	0.699±0.007	0.695±0.006	0.682±0.007
BERT	Oversampling	MAIA	0.644±0.002	0.648±0.006	0.627±0.003
BERT	Web scraping	MAIA	0.644±0.005	0.650±0.004	0.634±0.005
BERT	Combination	MAIA	0.614±0.010	0.621±0.009	0.601±0.010

5.2.3 Hierarchical classification

The local- and hierarchical classification results for the SVM and BERT models are shown in Table 14. The models were trained using non-enriched data, i.e. normal training data, and training data using combined data enrichment. The classification performance was evaluated using macro- and weighted scores to emphasize the differences between the minority- and majority class labels. The local classification results consider the average scores for the local classification of the parent class labels. A detailed overview of local model performances can be found in Appendix B. Considering the average local classification scores in Table 14 we observe that both models perform similar on the local classification tasks.

Data enrichment has a small influence on the model performance. SVM show slightly better performance using normal training data, while BERT show most improvement after data enrichment. In addition, the differences between the macro and weighted scores suggest that both models perform better on the majority classes according to the large weighted performance scores.

Furthermore, the hierarchical results show that the models perform similar on both the normal- and the enriched training data. By comparing the hierarchical results with the flattened classification results for the child class in Table 14, we observe that both models show significantly improvement using normal training data. Hierarchical classification does not improve the classification performance of SVM using data enrichment. SVM show better classification performance using flattened classification with data enrichment on the child class, which is remarkable.

Table 14
Performance of the SVM and BERT models for local- and hierarchical classification on the test set. The models are trained using non-enriched training data, i.e. Normal, and a combination of web scraping and oversampling. The local classification performance corresponds to the average scores of the local models for the parent class labels. The hierarchical scores corresponds to the hierarchical performance metric discussed in Section 4.4.6. The results consider the macro- and weighted-averaging performance scores.

Local classification

		Macro			Weighted		
Model	Type	P	R	F1	P	R	F1
SVM	Normal	0.673	0.691	0.670	0.801	0.811	0.794
SVM	Combi	0.664	0.690	0.663	0.803	0.806	0.791
BERT	Normal	0.642±0.042	0.674±0.046	0.641±0.043	0.781±0.022	0.799±0.020	0.777±0.019
BERT	Combi	0.676±0.027	0.702±0.037	0.675±0.029	0.796±0.016	0.804±0.014	0.790±0.014

Hierarchical classification

		Macro			Weighted		
Model	Type	P	R	F1	P	R	F1
SVM	Normal	0.793	0.766	0.769	0.871	0.877	0.867
SVM	Combi	0.770	0.787	0.767	0.886	0.885	0.878
BERT	Normal	0.765±0.003	0.766±0.003	0.755±0.002	0.854±0.016	0.862±0.017	0.851±0.013
BERT	Combi	0.784±0.018	0.772±0.018	0.768±0.023	0.878±0.009	0.882±0.008	0.874±0.008

5.2.4 Generalization LOO set

Generalizing performance of the optimized SVM and BERT models was tested using a leave-one-out (LOO) company as test set while training the models with data according to the other companies, i.e. Client IDs. We used the data with combined data enrichment as training data. The results are shown in Table 15. We observe a significant difference between the macro- and weighted-averaging scores. The overall poor macro scores suggest that most misclassifications were caused by the minority classes. In addition, we observe that both SVM and BERT models perform well on the majority classes according to the greater weighted performance scores. There is a company that show poor results on both

macro- and weighted classification scores. Client ID 2020_046 contains a company specific product class that was not present during training. The poor performance suggest that both SVM and BERT were not able to predict new classes in the test set that were not seen during training.

Table 15
Performance flattened SVM and BERT classification for parent class AC on LOO company as test set. Classifiers were trained using the combination enriched data while leaving one client (Client ID) out of training. The performance metrics describing both macro- and weighted-averaging scores, M and W, respectively.

		Precision		Recall		F1	
Client ID	Model	M	W	M	W	M	W
2021_015	SVM	0.468	0.940	0.304	0.940	0.354	0.931
2021_015	BERT	0.257	0.953	0.246	0.940	0.239	0.942
2021_016	SVM	0.351	0.896	0.339	0.893	0.317	0.889
2021_016	BERT	0.313	0.926	0.399	0.901	0.329	0.911
2020_046	SVM	0.486	0.625	0.227	0.511	0.259	0.484
2020_046	BERT	0.318	0.497	0.194	0.344	0.207	0.377
2021_062	SVM	0.385	0.874	0.231	0.886	0.272	0.868
2021_062	BERT	0.445	0.879	0.305	0.884	0.347	0.872
2020_075	SVM	0.454	0.741	0.282	0.753	0.315	0.698
2020_075	BERT	0.364	0.775	0.286	0.749	0.298	0.701

Client ID: 2020_046 contains a company specific product class, parent class label 13, i.e. Software. This product class was only present in the test set and was not in the training data, i.e. the data from the other companies.

5.3 Error Analysis

In this chapter we conduct an error analysis on the product classes that show a high number of misclassifications according to the local classification results in Section 5.2.3. We will illustrate incorrect predicting behavior for the classification models using an example. In addition, we examine the most important features and attention weights for the SVM and BERT models, respectively.

5.3.1 Model misclassifications

Table 16 shows a selection of misclassified product classes by considering local SVM models with macro scores < 0.75. We observed that parent class Voedsel en dranken showed the most misclassifications with 127 classification below a macro score of 0.75. Therefore, we decided to analyze a selection of product descriptions and related predictions for this class.

Table 16
Selection of local SVM classification results corresponding to macro scores < 0.75 using training data with combined data enrichment. The results illustrate the product classes by the number of miclassifications, i.e. Counts.

Parent class	Counts	Precision_M	Recall_M	F1_M
18 Voedsel en dranken	127	0.534	0.567	0.512
1 Diervoer	51	0.400	0.500	0.444
13 Software	29	0.250	0.500	0.333

Table 17 shows a sample of products for parent class Voedsel en dranken. We observed that both classification models interpret the word ananas, i.e. pineapple differently whenever the context changes. Comparisons of the corresponding misclassifications show that some incorrect predicted labels are actually correct due to subjective manual labeling. We suggest that there were several human annotators involved, resulting from product descriptions labeling with both pineapple or fruit (mixed). Moreover, we found that incorrect classifications are caused by wrong manual labeling. Product descriptions as "minute maid jus orange" are classified with product label "Nut", while it is evidently considered to be classified as Juice. Table 18 shows a selection of products descriptions that contains the words "jus orange" and varying incorrect child class labels. The results show incorrectly labeled product descriptions and suggest that most of these products cause incorrect classification models. To illustrate the incorrect predicting behavior we examined the SVM features and the BERT attention weights.

5.3.2 Feature importance and attention

In order to illustrate the most important tokens for the classification of the product classes, we have ranked the SVM features for the prediction of the product class labels using the model coefficients. The token importance for BERT can be assessed considering the attention weights that reflects the influence of tokens on classification performance. To illustrate the latter, we examined a selection of product classes that showed poor performance according to the local classification models with combined data enrichment in Appendix B. Table 19 shows the most important word features according to the ten largest SVM coefficients

Table 17
Selection of products for parent class label Voedsel en dranken. The results show the predicted child class labels following the local classification models using combined data enrichment.

Omschrijving	True label	SVM	BERT
del monte ananasschijven op sap	Fruit (mixed)	Fruit (mixed)	Pineapple
grand gerard ananas stukjes	Fruit (mixed)	Pineapple	Fruit (mixed)
fruitlife framboos gruis	Fruit (mixed)	Raspberry	Raspberry
minute maid jus orange	Juice	Nut	Nut
ananas bio	Pineapple	Pineapple	Pineapple

Table 18
Example of incorrect child class labeling for product descriptions containing the words "jus orange".

Omschrijving	AnalyseCcategory	MAIA KeyProduct
minute maid jus orange	Voedsel en dranken	Juice
minute maid jus orange	Voedsel en dranken	Nut
jus orange liter	Voedsel en dranken	Orange juice
royal club jus orange	Voedsel en dranken	Cola

and BERT attention weights for the product classes.

Considering the SVM feature importance for the Software class, we observe that the important tokens show general words rather that ones specifically related to the classes. The BERT results show that words with the most attention corresponds to out-of-vocabulary words, i.e. sub-words indicated with prefix "##". Sub-words have the tendency to be general rather than class specific and the attention on these tokens most likely result in a poor classification model. In addition, Figure 4 illustrates an example of a product description and the attention weights on the sentence tokens. BERT appears to concentrate attention to words that are non-related to software products. For example, we observe that the word 'online' is negatively related to the software class, while this word relates the strongest positive coefficient according to the SVM classifier.

Furthermore, the results show that BERT fails in classifying product descriptions related to the class Diervoer, i.e. Animal feed, while SVM show good classification performance according to the weighted F1-scores. The most important tokens for the BERT model consists mostly of out-of-vocabulary words. The results suggest that out-of-vocabulary words have the tendency to decrease the classification performance of BERT models for short product descriptions.

Table 19
Feature importance and attention weights for the local SVM and BERT models. The models were trained using combined data enrichment. The results show the most important tokens for the classification of the product classes. The SVM feature importance show the most important tokens related to the largest coefficients. BERT attention is described as the average attention for most important tokens.

Model	Class	$F1_M$	$F1_W$	Word features
SVM	Diervoer	0.444	0.711	'masters', 'vlasstrooisel', 'tarwestro', 'weidehooi', 'pianissimo', 'linamix', 'lucerne', 'elektrolytenslobber', 'nutramon', 'hondenbrokken'
SVM	Software	0.333	0.333	'online', 'abonnement', 'online abonnement', 'online learning', 'learning', 'per', 'agreement', 'learning agreement', 'marinetraffic', 'sen'
SVM	Voertuigen	0.633	0.818	'vw', 'opel', 'renault', 'peugeot', 'cateringtruck', 'bmw', 'pd', 'volvo', 'pb', 'mercedes'
BERT	Diervoer	0.111±0.096	0.056±0.051	'omega', 'gras', 'luce', '##ssimo', '##x', '##elen', '##oi', '##kken', '##mm', 'standard'
BERT	Software	0.333±0.000	0.333±0.000	'agreement', 'engels', 'services', '##ment', '##it', '##op', 'premium', 'media', 'user', 'gideon'
BERT	Voertuigen	0.279±0.070	0.504±0.034	'bmw', 'pd', 'peugeot', 'ford', 'black', 'golf', 'citroen', 'black', 'trail', 'fs'

[CLS] gideon sim users online abo ##nne ##ment bt ##w aan ##passing door ##ge ##voerd in het kader van spirit [SEP]

Figure 4. Illustration of the BERT attention weights considering a product description corresponding to the parent class Software. The color intensity reflects the strength of the attention, in which green and red colors corresponds to positive- and negative effect on classification performance, respectively.

6 Discussion

In this section we will discuss the results of this research in the context of the research questions. We will summarize our main findings and interpreting the results. In addition, we will reflect on the limitations.

6.1 Interpretations

The experiments showed remarkable outcomes and unexpected classification results. Our first experiment was considering the effects of hyperparameter optimization on the classification performances of the SVM and BERT models. According to the SVM classification experiments, the parent class and the child class models show significant improvement on the classification performance after hyperparameter optimization. Both models considered BOW and TF-IDF as optimal feature representation to improve the classification perfor-

mance. The child class model incorporated additional Bigrams. Because of the large set of minority classes with a relatively small number of related product descriptions we suspect that the child class model improved classification performance by including additional word features. Furthermore, we observed that the optimization primarily increased the recall scores of the parent class models. We suggest that the SVM decision boundaries are sensitive to imbalanced data and have the tendency to be skewed towards the minority classes, resulting in more false negatives. We expect that L2-regularization will overcome this problem by reducing the skewness of the decision boundaries and decreasing the number of false negatives.

Considering the BERT models, we observed that the effect of hyperparameter optimization on the classification performance was minimal. We used dropout regularization on both the hidden- and the classification layers. The results suggest that employing dropout regularization has only a small effect on imbalanced data with with a large number of minority classes that considers short product descriptions. We suggest that dropout is most effective for the majority classes. This is because these classes contain a sufficient number of training samples. Performing dropout regularization on minority classes caused a loss of information in a setting where there was already insufficient training data available for these classes. We expect that this was the result of the small increase in model performance after hyperparameter optimization.

We trained and tested the classification models after conducted data enrichment on the training data to tread for class imbalance. The experiment showed that the flattened classification of both models using the parent class showed minimal improvement on the model performances, e.g. oversampling decreased the model performance. We expect that oversampling resulted in overfitting the training data by duplicating less representative examples and causing a bias towards these minority classes. Contrarily, web scraping also increased the minority classes but used new product variations from web results and increased the classification performance for the SVM models. Furthermore, considering the child class data, we observed that the SVM classification models show a significant improvement using data enrichment. The results suggest that SVM performs well on imbalanced data with a large collection of minority classes that have been treated with data enrichment. This corresponds with prior research by Joachims (1998) and follows Cover's theorem (Cover, 1965) that emphasized the linearly separability of such data. Contrarily, the BERT models showed poor performance according to the child class training data with data enrichment. We expect that the loss function puts too much weight on the minority classes and is learning from a small training set with possible non-discriminating features resulting in poor generalizing abilities on the test set. In addition, the error analysis of the BERT models showed that poor classification performance exhibits tokens that were out-of-vocabulary and had the tendency to be general rather than class specific tokens. We observed that the attention weights were focused too much on the general or nonrepresentative tokens. We suggest that conducting data enrichment on classes containing high number of such tokens most likely results in an increased chance for overfitting on the training data.

As last, the hierarchical classification results showed that BERT performs significantly better by decreasing the set of (minority) classes. This is what we expected, since the flatten classification results suggested that BERT loss function was biased towards the minority classes. Decreasing the size of minority classes results in a better chance for classifying the correct class. However, we observed that SVM showed better performance on the flattened

classification of the child class data using data enrichment. This result suggest that SVM performed better using a large set of imbalanced labels rather than a smaller set of imbalanced labels using data enrichment. The difference is that when using a large set of classes, the vocabulary size tends to be large, resulting in a high-dimensional feature representation. A small set of labels, consisting of a smaller vocabulary size, results in less sparse representation of features. We suggest that SVM is better able to perform linear classification for enriched imbalanced data in a high-dimensional feature setting rather than a low-dimensional, i.e. less sparse representation of features. However, this is something that is interesting to look into further.

6.2 Limitations

There were several complications to constructing an unbiased training set using the data enrichment methods. Each classification model was trained using the enriched training set while keeping the test set fixed. Enriching the training data during the classification experiments was not efficient because of the computation time. The web scraping procedure took approximately 6 or 7 hours to collect web data and process the results using sentence-BERT with cosine-similarity for prior-filtering the text for semantic similarity. Therefore, we considered the web scraping prior to the classification experiments and conclude that changing the training data during the classification experiments is not possible.

The error analysis illustrated that misclassifications can be addressed to subjective or incorrect manual labeling. It is questionable if other product descriptions are actually misclassified or suffer from incorrect manual labeling. If so, the classification performances of our experiments may be underestimated. In addition, the results have indicated that product classes with a large number of misclassification are characterized by words that are not related to the product class, i.e. non-discriminating tokens. The results showed that the BERT attention weights had the tendency to focus on general words rather than class specific. Moreover, these tokens considered out-of-vocabulary tokens corresponding to sub-words that are less representative for the product class.

The BERT classification models were optimized using the normal training data. Optimization of such complex and deep models takes a substantial amount of training time. To illustrate, training BERT on the parent class takes approximately 800 seconds using the largest proposed learning rate of $5e^{-5}$ with a TPU machine. Using a grid space of hyperparameters for three training- and model parameters, each considering 4 options, results in $800 * 3 * 4 = 9600$ seconds. Due to its stochastic character, we repeated each training 5 times to construct unbiased estimates on classification performance. As a result, optimizing a single BERT model for the child class took $48,000$ seconds corresponding to approximately 13 hours of training. This large training time is not result in significantly better classification performance w.r.t. the less complex and time efficient SVM classification models. In addition, the TPU memory failed to clear memory after several runs and required a hard reset resulting in loss of previous training data. We have not found a proper solution for this memory problem. Moreover, long training times require a substantial amount of energy, which makes classification with BERT highly unsustainable.

7 Conclusion

This research was motivated by the necessity to encounter the imbalanced- and hierarchically ordered class labels for short product descriptions to perform impact assessments. We experimented with SVM and BERT models and utilize both models as a flattened- and a hierarchical multiclass classification tasks. In addition, we used two data enrichment methods to treat the imbalance of product classes. Consequently, after performing all experiments and evaluating the results we answer the following research questions:

How can product data with hierarchically ordered and imbalanced classes be effectively classified?

This main research question can be answered with the following subquestions:

How can we treat data imbalance to increase the quality of classifications?
We proposed two data enrichment methods to treat the class imbalance in the training data, oversampling and a web scraping method. The results showed that data enrichment was not improving the BERT models for flattened classification tasks, it did improved the hierarchical classification model. The flattened SVM models showed improved performance after data enrichment, except for the parent class using oversampling. The hierarchical SVM classification model show similar results with respect to training data without data enrichment.

How can we address the hierarchically oriented product classes?
In order to address the class hierarchy we used two classification approaches. Flattened multiclass classification and hierarchical multiclass classification. The hierarchical classification corresponded to recursively training the classification models per parent class by using local information. This hierarchical approach reduced the large set of child labels into several small sets. The results showed that hierarchical classification improves classification performance for the BERT models. The SVM models show minimal improvement using the hierarchical classification approach.

How do deep learning methods differ from conventional learning methods for text classification?
In this research we experimented with SVM and BERT, a conventional- and a deep learning method, respectively. SVM is a deterministic classification method that was considering a BOW representation, a sparse representation of words. This representation method does not account for word meaning or positioning of words in a product description. BERT is stochastic learning method that is pre-trained using a large text corpus and learns the meaning of words and word positions in a self-supervised way. The pre-trained models can be used in a transfer learning setting and be used for downstream tasks. We considered a multilingual BERT model that was fine-tuned for the classification of product descriptions using the product data.

What makes the classification methods effectively classifying the product descriptions and what not?
According to the experiments and the error analysis we observed that models with a large number of misclassification were focused on general words rather than ones specifically

related to a product class. These non-discriminating features have resulted in poor generalizing abilities on the test data. Contrarily, product descriptions that contained more product specific tokens improved the model performance.

In addition, we observed that SVM classification models had the tendency to perform well on data containing a large set of imbalanced classes, i.e. many minority classes. BERT show better classification performance by using a hierarchical classification approach and reducing the number of (imbalanced) labels. We suggest that the different classification behavior with presence of many minority classes is related to the differences in the loss function characteristics of the classification models.

7.1 Future work

The error analysis showed that the misclassified product classes considered model features that were related to out-of-vocabulary tokens. We observed that most of these product descriptions considered sub-words rather than product specific words. To improve classification performance it would be advantageous to pre-train BERT models on domain specific data related to products. Furthermore, it would be interesting to further develop our proposed web scraping method. The search space could be extended with different data sources, e.g. e-commerce websites or databases. In addition, the web scraping method can be used to incorporate new product classes or to construct complete new datasets.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from `http://tensorflow.org/`

Ahn, L., Blum, M., Hopper, N., & Langford, J. (2003). Captcha: using hard AI problems for security. *Advances in Cryptology*, *2656*, 294-311. doi: 10.1007/3-540-39200-9_18

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. Retrieved from `https://arxiv.org/abs/1206.5533` doi: 10.48550/ARXIV.1206.5533

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer-Verlag. doi: 10.5555/1162264

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. Retrieved from `https://arxiv.org/abs/1607.04606` doi: 10.48550/ARXIV.1607.04606

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi: 10.1613/jair.953

Chen, J., Hu, Y., Liu, J., Xiao, Y., & Jiang, H. (2019). Deep short text classification with knowledge powered attention. Retrieved from `https://arxiv.org/abs/1902.08050` doi: 10.48550/ARXIV.1902.08050

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, *EC-14*(3), 326-334. doi: 10.1109/PGEC.1965.264137

Devlin, J. (2021). *Stanford cs224n: Nlp with deep learning | winter 2020 | bert and other pre-trained language models*. Retrieved from `https://www.youtube.com/watch?v=knTc-NQSjKA&t=1546s&ab_channel=StanfordOnline`

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. Retrieved from `https://arxiv.org/abs/1810.04805` doi: 10.48550/ARXIV.1810.04805

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch bert model. Retrieved from `https://arxiv.org/abs/1912.09582` doi: 10.48550/ARXIV.1912.09582

D'hondt, E., Verberne, S., Oostdijk, N., & Boves, L. (2017). Patent classification on subgroup level using balanced winnow. , 299-324. doi: 10.1007/978-3-662-53817-3_11

Feng, S., Fu, P., & Zheng, W. (2018). A hierarchical multi-label classification method based on neural networks for gene function prediction. *Biotechnology & Biotechnological Equipment*, *32*(6), 1613-1621. Retrieved from `https://doi.org/10.1080/13102818.2018.1521302` doi: 10.1080/13102818.2018.1521302

Fernández, A., García, S., Galar, M., Prati, R., Krawczyk, B., & Herrera, F. (2018). *Learning from imbalanced data sets*. doi: 10.1007/978-3-319-98074-4

Gao, P., Zhao, J., Ma, Y., Tanvir, A., & Jin, B. (2022). Hft-onlstm: Hierarchical and fine-tuning multi-label text classification. Retrieved from `https://arxiv.org/abs/2204.08115` doi: 10.48550/ARXIV.2204.08115

Ghoroghi, A., Rezgui, Y., Petri, I., & Beach, T. (2022). Advances in application of machine learning to life cycle assessment: a literature review. *The International Journal of Life Cycle Assessment*, *27*, 1-24. doi: 10.1007/s11367-022-02030-3

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. New York, NY, USA: Springer New York Inc.

He, H., & Ma, Y. (2013). Class imbalance learning methods for support vector machines. In *Imbalanced learning: Foundations, algorithms, and applications* (p. 83-99). doi: 10.1002/9781118646106.ch5

Huang, K., Hussain, A., Wang, Q., & Zhang, R. (2019). *Deep learning: Fundamentals, theory and applications*. Springer. Retrieved from https://books.google.nl/books?id=IA4mxQEACAAJ

IBM. (2020). *Natural language processing (nlp)*. Retrieved 2022-07-21, from https://www.ibm.com/cloud/learn/natural-language-processing

Jahanshahi, H., Ozyegen, O., Cevik, M., Bulut, B., Yiğit, D., Gonen, F. F., & Basar, A. (2021). Text classification for predicting multi-level product categories. *ArXiv*, *abs/2109.01084*.

Jinfeng, L., Shouling, J., Tianyu, D., Bo, L., & Ting, W. (2019). TextBugger: Generating adversarial text against real-world applications. doi: 10.14722/ndss.2019.23138

Joachims, T. (1998). Text categorization with support vector machines. *Proc. European Conf. Machine Learning (ECML'98)*. doi: 10.17877/DE290R-5097

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., … Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. Retrieved from https://arxiv.org/abs/1704.04760 doi: 10.48550/ARXIV.1704.04760

Jurafsky, D., & Martin, J. H. (2021). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (1st ed.). USA: Prentice Hall PTR.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. Retrieved from https://arxiv.org/abs/1609.04836 doi: 10.48550/ARXIV.1609.04836

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. Retrieved from https://arxiv.org/abs/1412.6980 doi: 10.48550/ARXIV.1412.6980

Kiritchenko, S., & Famili, F. (2005). Functional annotation of genes using hierarchical text categorization. *Proceedings of BioLink SIG, ISMB*. Retrieved from https://www.site.uottawa.ca/~stan/papers/2004/p15.pdf

Martinez-Gomez, P., Papachristoudis, G., Blauvelt, J., Rachlin, E., & Simhon, S. (2021). Enhancement and analysis of tars few-shot learning model for product attribute extraction from unstructured text. Retrieved from https://assets.amazon.science/10/72/e3dcf5174fcdb724a51b492c1fc4/enhancement-and-analysis-of-tars-few-show-learning-model-for-product-attribute-extraction-from-unstructured-texts.pdf

Metabolic. (2022). *Metabolic company profile*. Retrieved 2022-07-18, from https://www.metabolic.nl/about

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Retrieved from https://arxiv.org/abs/1301.3781 doi: 10.48550/ARXIV.1301.3781

More, A. (2016). Attribute extraction from product titles in ecommerce. Retrieved from https://arxiv.org/abs/1608.04670 doi: 10.48550/arXiv.1608.04670

Oksanen, J., Cocarascu, O., & Toni, F. (2021). Automatic product ontology extraction from textual reviews. Retrieved from https://arxiv.org/abs/2105.10966 doi: 10.48550/ARXIV.2105.10966

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. doi: 10.5555/1953048.2078195

Pennington, J., Socher, R., & Manning, C. (2014, October). GloVe: Global vectors for word representation. , 1532–1543. Retrieved from https://aclanthology.org/D14-1162 doi: 10.3115/v1/D14-1162

Pörtner, H.-O., Roberts, D., Tignor, M., Poloczanska, E., Mintenbeck, K., Alegría, A., … Rama, B. (2022). *Climate change 2022: Impacts, adaptation, and vulnerability.* Retrieved from https://www.ipcc.ch/report/ar6/wg2/

Qaiser, S., & Ali, R. (2018, 07). Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, *181*. doi: 10.5120/ijca2018917395

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. Retrieved from https://arxiv.org/abs/1908.10084

Richardson, L. (2007). Beautiful soup documentation. Retrieved from https://beautiful-soup-4.readthedocs.io/en

Rued, S., Ciaramita, M., Mueller, J., & Schuetze, H. (2011). Piggyback: Using search engines for robust cross-domain named entity recognition. , 965–975. Retrieved from http://www.aclweb.org/anthology/P/P11/P11-1097.pdf

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. Retrieved from https://arxiv.org/abs/1910.01108 doi: 10.48550/ARXIV.1910.01108

Sharma, N., Verlekar, P., Ashary, R., & Zhiquan, S. (2017). Regularization and feature selection for large dimensional data. Retrieved from https://arxiv.org/abs/1712.01975 doi: 10.48550/ARXIV.1712.01975

Silla, C. N., & Freitas, A. A. (2010). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, *22*, 31-72.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. Retrieved from https://arxiv.org/abs/1803.09820 doi: 10.48550/ARXIV.1803.09820

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune bert for text classification? Retrieved from https://arxiv.org/abs/1905.05583 doi: 10.48550/ARXIV.1905.05583

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. Retrieved from https://arxiv.org/abs/1706.03762 doi: 10.48550/ARXIV.1706.03762

Weiss, G. (2004). Mining with rarity: A unifying framework. *SIGKDD Explorations*, *6*, 7-19. doi: 10.1145/1007730.1007734

Wernet, G., Bauer, C., Steubing, B., Reinhard, J., Moreno-Ruiz, E., & Weidema, B. (2016, April 21). The ecoinvent database version 3 (part i): overview and methodology. *International Journal of Life Cycle Assessment*, *21*(9), 1218–1230. doi: 10.1007/s11367-016-1087-8

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., … Rush, A. M. (2019).

Huggingface's transformers: State-of-the-art natural language processing. Retrieved from https://arxiv.org/abs/1910.03771 doi: 10.48550/ARXIV.1910.03771

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., … Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. Retrieved from https://arxiv.org/abs/1609.08144 doi: 10.48550/ARXIV.1609.08144

Yu, H.-F., Ho, C.-H., Arunachalam, P., Somaiya, M., & Lin, C.-J. (2012). Product title classification versus text classification. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/papers/title.pdf

Appendix

A BERT Results

A.1 Flattened classification

Table 20
Grid search on different Learning Rates. Model: bert-base-uncased (bert-base). Response: parent class AC.

Model	Batch size	Learning rate	Precision$_{macro}$	Recall$_{macro}$	F1-score$_{macro}$	t(s)
bert-base	128	1e-5	0.907±0.027	0.761±0.055	0.809±0.046	671±64
bert-base	128	2e-5	0.885±0.036	0.755±0.008	0.795±0.018	485±33
bert-base	128	3e-5	0.837±0.016	0.762±0.023	0.789±0.020	466±28
bert-base	128	4e-5	0.901±0.018	0.796±0.022	**0.829±0.015**	461±39
bert-base	128	5e-5	0.878±0.027	0.788±0.021	0.815±0.014	437±21

Table 21
Grid search on different Learning Rates. Model: bert-base-multilingual (bert-multi). Response: parent class AC.

Model	Batch size	Learning rate	Precision$_{macro}$	Recall$_{macro}$	F1-score$_{macro}$	t(s)
bert-multi	128	1e-5	0.872±0.043	0.736±0.073	0.773±0.066	534±54
bert-multi	128	2e-5	0.895±0.027	0.774±0.045	0.809±0.042	519±27
bert-multi	128	3e-5	0.893±0.020	0.776±0.021	0.813±0.015	475±10
bert-multi	128	4e-5	0.890±0.024	0.812±0.030	**0.832±0.030**	460±27
bert-multi	128	5e-5	0.894±0.040	0.757±0.031	0.801±0.034	439±16

Table 22
Grid search on different Learning Rates. Model: bertje. Response: parent class AC.

Model	Batch size	Learning rate	Precision$_{macro}$	Recall$_{macro}$	F1-score$_{macro}$	t(s)
bertje	128	1e-5	0.888±0.040	0.743±0.036	0.789±0.040	612±88
bertje	128	2e-5	0.912±0.020	0.786±0.032	0.828±0.026	504±7
bertje	128	3e-5	0.908±0.034	0.799±0.022	**0.830±0.028**	473±26
bertje	128	4e-5	0.901±0.029	0.782±0.013	0.822±0.008	434±13
bertje	128	5e-5	0.885±0.040	0.769±0.040	0.804±0.040	455±18

Table 23
Grid search on different Learning Rates. Model: distilbert. Response: parent class AC.

Model	Batch size	Learning rate	Precision_{macro}	Recall_{macro}	F1-score_{macro}	t(s)
distilbert	128	1e-5	0.876±0.036	0.759±0.031	0.796±0.029	720±53
distilbert	128	2e-5	0.876±0.029	0.773±0.031	0.807±0.025	526±17
distilbert	128	3e-5	0.901±0.037	0.776±0.034	**0.815±0.028**	493±14
distilbert	128	4e-5	0.865±0.022	0.773±0.026	0.784±0.026	429±14
distilbert	128	5e-5	0.851±0.038	0.773±0.034	0.795±0.020	439±19

Table 24
Grid search on different Learning Rates. Model: bert-base-multilingual (bert-multi). Response: child class MAIA.

Model	Batch size	Learning rate	Precision_{macro}	Recall_{macro}	F1-score_{macro}	t(s)
bert-multi	128	1e-5	0.611±0.019	0.605±0.016	0.592±0.018	811±28
bert-multi	128	2e-5	0.677±0.012	0.681±0.015	0.664±0.014	819±23
bert-multi	128	3e-5	0.695±0.010	0.702±0.006	0.682±0.007	812±25
bert-multi	128	4e-5	0.693±0.018	0.698±0.019	0.680±0.019	815±38
bert-multi	128	5e-5	0.696±0.022	0.705±0.024	**0.685±0.021**	804±33

B Hierarchical classification

Table 25
Results local classification by the optimized SVM and BERT models on test set using the normal training data. Models were trained using local information per parent class C_p with the corresponding child class labels. The classification performance is described using both macro- and weighted-averaging scores to illustrate the difference between minority- and majority class predictions. The number of training samples per parent class are illustrated by N.

		SVM						BERT					
		Weighted			Macro			Weighted			Macro		
C_p	N	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	441	0.884	0.940	0.911	0.387	0.400	0.393	0.931±0.018	0.924±0.030	0.926±0.016	0.388±0.000	0.435±0.000	0.399±0.000
1	39	0.640	0.800	0.711	0.400	0.500	0.444	0.640±0.000	0.800±0.000	0.711±0.000	0.400±0.079	0.500±0.148	0.444±0.098
2	258	0.948	0.931	0.931	0.833	0.833	0.822	0.769±0.028	0.828±0.024	0.791±0.015	0.634±0.028	0.685±0.060	0.654±0.040
3	1,025	0.827	0.833	0.824	0.631	0.679	0.646	0.816±0.028	0.826±0.022	0.814±0.025	0.572±0.069	0.610±0.061	0.580±0.064
4	2,038	0.933	0.921	0.919	0.890	0.874	0.868	0.916±0.005	0.914±0.007	0.908±0.005	0.830±0.023	0.810±0.013	0.806±0.016
5	1,916	0.917	0.915	0.910	0.858	0.871	0.857	0.882±0.008	0.885±0.007	0.877±0.008	0.768±0.016	0.790±0.009	0.763±0.010
6	244	0.750	0.750	0.750	0.559	0.559	0.559	0.730±0.045	0.814±0.030	0.759±0.036	0.581±0.037	0.672±0.039	0.611±0.040
7	4,531	0.993	0.992	0.992	0.984	0.984	0.982	0.993±0.002	0.993±0.001	0.993±0.002	0.989±0.018	0.976±0.016	0.981±0.017
8	112	1.000	1.000	1.000	1.000	1.000	1.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
9	975	0.883	0.862	0.857	0.652	0.673	0.644	0.892±0.012	0.861±0.014	0.861±0.013	0.681±0.042	0.701±0.046	0.673±0.041
10	909	0.883	0.901	0.885	0.690	0.657	0.655	0.913±0.012	0.921±0.000	0.907±0.004	0.732±0.045	0.690±0.033	0.685±0.033
11	733	0.977	0.988	0.982	0.945	0.950	0.948	0.973±0.007	0.978±0.005	0.975±0.006	0.865±0.035	0.884±0.041	0.871±0.036
12	913	0.944	0.922	0.927	0.770	0.769	0.763	0.928±0.008	0.910±0.011	0.907±0.009	0.799±0.040	0.770±0.041	0.773±0.041
13	29	0.250	0.500	0.333	0.250	0.500	0.333	0.250±0.000	0.500±0.000	0.333±0.000	0.250±0.000	0.500±0.000	0.333±0.000
14	198	0.859	0.870	0.856	0.854	0.861	0.852	0.946±0.017	0.904±0.036	0.913±0.032	0.910±0.053	0.896±0.059	0.894±0.062
15	338	0.790	0.842	0.809	0.652	0.688	0.661	0.841±0.038	0.895±0.046	0.859±0.049	0.659±0.057	0.699±0.092	0.668±0.078
16	22	0.000	0.000	0.000	0.000	0.000	0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
17	137	0.938	0.812	0.865	0.714	0.638	0.670	0.706±0.162	0.700±0.120	0.670±0.126	0.604±0.268	0.649±0.199	0.588±0.226
18	17,128	0.772	0.747	0.747	0.674	0.674	0.655	0.746±0.008	0.748±0.005	0.737±0.006	0.627±0.009	0.648±0.004	0.620±0.005
19	93	0.818	0.818	0.818	0.633	0.633	0.633	0.585±0.038	0.509±0.050	0.504±0.034	0.328±0.046	0.289±0.091	0.279±0.070
20	54	0.857	0.714	0.690	0.875	0.833	0.792	0.905±0.000	0.857±0.000	0.857±0.000	0.917±0.000	0.917±0.000	0.900±0.000
21	404	0.755	0.778	0.748	0.557	0.629	0.559	0.816±0.045	0.818±0.040	0.800±0.042	0.581±0.067	0.657±0.053	0.588±0.058
Average		0.801	0.811	0.794	0.673	0.691	0.670	0.781±0.022	0.799±0.020	0.777±0.019	0.642±0.042	0.672±0.046	0.641±0.043

Table 26
Results local classification by the optimized SVM and BERT models on test set using the combined data enrichment on the training data. Models were trained using local information per parent class C_p with corresponding related child class labels. The classification performance is described using both macro- and weighted-averaging scores to illustrate the difference between minority- and majority class predictions. The number of training samples per parent class are illustrated by N.

		SVM						BERT					
		Weighted			Macro			Weighted			Macro		
C_p	N	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	1,144	0.920	0.940	0.930	0.330	0.333	0.331	0.927±0.030	0.933±0.012	0.930±0.020	0.328±0.046	0.329±0.043	0.328±0.045
1	51	0.640	0.800	0.711	0.400	0.500	0.444	0.036±0.034	0.133±0.115	0.056±0.051	0.070±0.061	0.278±0.255	0.111±0.096
2	469	0.948	0.931	0.931	0.833	0.833	0.822	0.866±0.002	0.897±0.000	0.868±0.001	0.778±0.003	0.821±0.000	0.784±0.002
3	1,743	0.862	0.868	0.861	0.744	0.779	0.754	0.877±0.012	0.868±0.018	0.864±0.014	0.750±0.080	0.761±0.073	0.746±0.077
4	3,920	0.919	0.899	0.896	0.835	0.789	0.790	0.939±0.002	0.934±0.000	0.930±0.003	0.868±0.001	0.872±0.011	0.855±0.001
5	3,029	0.865	0.864	0.859	0.733	0.756	0.735	0.899±0.014	0.905±0.010	0.895±0.011	0.816±0.018	0.836±0.018	0.812±0.017
6	485	0.839	0.857	0.830	0.767	0.817	0.771	0.889±0.003	0.929±0.000	0.903±0.001	0.807±0.006	0.867±0.000	0.828±0.005
7	9,410	0.991	0.990	0.990	0.984	0.972	0.975	0.994±0.000	0.994±0.000	0.994±0.000	0.998±0.000	0.986±0.000	0.991±0.000
8	145	1.000	1.000	1.000	1.000	1.000	1.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
9	1,462	0.907	0.890	0.885	0.717	0.749	0.713	0.917±0.006	0.887±0.014	0.890±0.011	0.716±0.032	0.726±0.046	0.704±0.036
10	2,101	0.894	0.891	0.882	0.554	0.512	0.505	0.942±0.019	0.937±0.015	0.929±0.018	0.773±0.035	0.706±0.043	0.712±0.038
11	1,300	0.977	0.988	0.982	0.945	0.950	0.948	0.986±0.004	0.988±0.000	0.986±0.002	0.912±0.012	0.920±0.026	0.914±0.016
12	1,785	0.919	0.892	0.898	0.712	0.702	0.700	0.953±0.010	0.935±0.006	0.940±0.007	0.803±0.024	0.793±0.018	0.794±0.021
13	32	0.250	0.500	0.333	0.250	0.500	0.333	0.250±0.000	0.500±0.000	0.333±0.000	0.250±0.000	0.500±0.000	0.333±0.000
14	279	0.880	0.870	0.868	0.801	0.795	0.794	0.967±0.029	0.942±0.050	0.948±0.045	0.945±0.052	0.939±0.064	0.936±0.060
15	624	0.790	0.842	0.809	0.652	0.688	0.661	0.833±0.026	0.868±0.026	0.843±0.028	0.636±0.012	0.666±0.047	0.639±0.034
16	29	0.000	0.000	0.000	0.000	0.000	0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
17	213	0.896	0.812	0.834	0.619	0.638	0.598	0.920±0.030	0.875±0.000	0.891±0.016	0.647±0.038	0.611±0.048	0.624±0.042
18	45,003	0.756	0.738	0.735	0.624	0.654	0.618	0.783±0.004	0.752±0.002	0.753±0.003	0.677±0.003	0.706±0.002	0.670±0.002
19	141	0.818	0.818	0.818	0.633	0.633	0.633	0.765±0.092	0.727±0.000	0.739±0.043	0.564±0.120	0.541±0.064	0.548±0.091
20	81	0.929	0.857	0.867	0.875	0.917	0.867	0.905±0.000	0.857±0.000	0.857±0.000	0.917±0.000	0.917±0.000	0.900±0.000
21	689	0.802	0.800	0.780	0.596	0.653	0.593	0.863±0.030	0.837±0.034	0.832±0.036	0.606±0.055	0.679±0.046	0.612±0.055
Average		0.809	0.820	0.804	0.664	0.690	0.663	0.796±0.016	0.804±0.014	0.790±0.014	0.676±0.027	0.702±0.037	0.675±0.029