Universiteit Leiden

Master Computer Science

Classification of bank transactions into multi-class, non-uniformly distributed ledger accounts

Name:	Thomas Vink
Student ID:	2043998
Date:	July 13, 2022
Specialisation:	Artificial Intelligence
1st supervisor:	Peter van der Putten
2nd supervisor:	Jan N. van Rijn
AFAS supervisor:	Erik van de Ven

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ABSTRACT In financial accounting, bank transactions need to be categorised into specific ledger accounts. Some bank transactions can be automatically categorised when specific identifying fields are available which can be used in combination with simple business rules. This is what we call structured data. However, a large set of transactions remain which do not carry sufficient structured data for automatic classification. Manual classification is quite a costly process. This thesis aims to automatically assign or suggest the correct ledger account whenever a new transaction is recorded in the system. This means solving a multi-class classification problem with unbalanced classes. To achieve a solution, a range of experiments will be run with various classical Machine Learning algorithms and a deep learning pre-processing algorithm. The final results show that a linear SVM model is able to reach 80% accuracy by suggesting the top-three classes. This depends on the number of features chosen as well as the number of instances available. The most useful pre-processing model of the data is a tf-idf transformer.

INDEX TERMS multi-class classification, SVM, text mining, word2vec

Table of Contents

I. INTRODUCTION

Most companies will record all transactions that happen in the company. Whether the transaction is a bank transaction or a financial transaction that belongs to its own company, accountants try to register it all for a complete overview of transactions done through the company. Considering bank transactions, the incoming and outgoing transactions will be classified into different ledger accounts. Ledger accounts contain the records of each bank transaction, categorised for convenience and as an overview of all the different transactions. Even if an employee buys lunch, it has to be recorded somehow. The accountants working for a specific company have to manually categorise the bank transactions, which takes time. Especially when large companies have lots of different ledger accounts to choose from.

Such a task can be partially solved through an algorithm that can accurately detect to which ledger account the new transaction belongs. These tasks are typically considered to be classification tasks.

The key idea of classification algorithms is to find an input where the algorithm can find relations between the input and the desired output. Considering we are able to use data that is already correctly labelled, we can narrow our algorithms down to supervised learning algorithms [8]. Supervised learning algorithms have labelled output such that the algorithm can learn which output could belong to which input. Since there are more than two different ledger accounts, the problem is considered to be a multi-class problem [1].

Every accountant can create as many ledger accounts as needed for their company. Therefore, the multi-class problem becomes expendable, meaning that new classes can be introduced at any time. Furthermore, some ledger accounts may be used frequently while others are used only on occasion. These two factors make the problem non-trivial. Moreover, the input columns consist of two different data type inputs. Either a structured input, such as the amount of the transaction, or free, sparse text input, such as the description of the transaction. 'BEA NR:619FGT 01.01.16/10.38 SHELL NIJKERK' is an example of a transaction description. Most well-known text classification algorithms only work well with rich text data, while such a small sentence is considered sparse text. The algorithm that will be used should find connections with limited data.

A classification task with such specific problems had been researched on invoices [4]. The classification of bank transactions into ledger accounts has also been researched with such problems by Ojala et al. (2018) [38]. However, Ojala et al. intentionally left out the description column as input for the machine learning algorithms. They made this choice based on the freeform text, which 'complicates the processing significantly'. Our research focuses on the description column. Therefore, this thesis provides novel research.

The research question we will focus on is as follows: "Is it possible to predict (with a certain probability) to which ledger account a bank transaction belongs by means of its unstructured data?" We will also explain our approach in detail, as most users for this particular problem will not have the background information they might require to be able to use the algorithm correctly. This research is carried out in cooperation with AFAS Software [46], a software as a service company for financial accounting.

The remainder of this thesis is structured as follows. Section II covers all research on the general topics related to our research. Section III explains the used algorithms. Section IV covers all experiments done with the various datasets and algorithms. Section V discusses the range of experiments and the results. Section VI lists all kinds of possibilities for future research and Section VII concludes.

II. RELATED WORK

The general problem of classification in financial software is a relatively untouched territory. Bardelli et al. [4] experimented with different algorithms and different pre-processing methods for automatic invoice classification. Even though invoice classification has different problems compared to bank transaction classification, some methods could be of use. The approach to choosing different pre-processing methods is a known practice in text classification [45] and could be of use in our research.

For bank transaction classification, Ojala et al. [38] discovered potential using different input columns from the bank transaction. Ojala et al. compared the accuracies for different companies on a trained model, revealing that one general trained model could work for different companies. However, Ojala et al. intentionally left out the description column because it contains free text, which is unstructured data. Nonetheless, a machine learning model can still learn from all the input if the words from the text are considered as features.

Both papers made use of several models. These need to be compared accordingly, while not ruling out any models prematurely. One of the most used comparison methods is the learning curve method [29] [41]. Full learning curves can be used to see which model is preferred depending on the number of training instances. An extension to the learning curves is the learning curve cross-validation (*LCCV*) [37]. This method uses so-called anchor points and discards candidates early whenever a method seems unlikely to be competitive. The anchor points are certain training sizes and cross-validation folds are used before the predictive performance is calculated on a separate test set. Based on the learning curves, one can determine which model performs better.

Different models are used with different classification problems. The multi-class datasets have a skewed outcome distribution (few classes appear frequently while most classes have few appearances) which makes the problem an imbalanced classification problem. Models such as support vector machines (*SVMs*) [50] and ensemble models [40] [33] have been researched for these problems. Undersampling [30] and adding weights or thresholds are well-known and used frequently in combination with these models for better performance with imbalanced classification problems.

Models can be either linear or non-linear. A comparison between linear and non-linear models [47] showed that non-linear models outperform linear models. However, the performance difference was insignificant when applied to most datasets, meaning that linear models might be beneficial for practical applications where an explanation of the model is also relevant.

All textual data needs to be transformed before models can use the data to train. A frequently used method is the 'term frequency inverse document frequency' (*tf-idf*) [23]. An extensive explanation of tf-idf will be given in Section III-A. tf-idf is also widely used with imbalanced classification problems [48] [3].

The models, or classifiers, can be separated into two different categories. Either soft or hard classifiers [51]. Soft classification, as stated by Liu et al. (2011) [32], 'generally estimates the class conditional probabilities explicitly and then makes the class prediction based on the largest estimated probability.' Liu et al. state that hard classification, on the other hand, 'directly estimates the classification boundary, bypassing the class probability estimation.'

Loads of classifiers have been created to solve text-mining-based classification problems. Naive Bayes [2] is a well-known, relatively straightforward classifier. Another well-known classifier is the support vector machine [50]. Furthermore, ensemble models, such as random forest [7] are also frequently used in text-mining-based classification problems. Xgboost [9], another ensemble model, is the current state-of-the-art machine learning model widely used [14]. In the deep learning department [36], BERT-like [13] algorithms are thriving (e.g. RoBERTa [31] or distilBERT [44]). The explanation of these algorithms will be given in Section III and Section VI.

III. METHODS

This section explains all algorithms that have been used. Before the algorithms can be explained, however, one must first have an understanding of classification algorithms in general. Furthermore, we divided our used algorithms into two categories, classical machine learning algorithms, and deep learning algorithms. Everything related to datasets is explained in Section IV.

A. CLASSIFICATION ALGORITHMS

Data mining has been researched comprehensively in studies and textbooks [49] [20]. According to D. Hand [20], 'Data mining is the discovery of interesting, unexpected or valuable structures in large datasets.' Data mining can be applied in a wide variety of sectors, e.g. bio-informatics [15], or education [42]. Data mining is a broad concept that consists of different applications and algorithms through the use of text data. Because the problem at hand consists of classification through sentences, the problem can be narrowed down to information extraction from text data [11] and supervised learning methods for text data [27], where both areas can be intertwined.

Supervised learning, the method to learn to correctly classify the input to a labelled output [8], needs to learn relations between the input and the label (or class) the input belongs to. Furthermore, if an algorithm can extract information from the input, our supervised learning algorithm might be able to make a more accurate prediction of which label belongs to which input. One problem that may arise with the supervised learning methods is the number of labels the dataset might contain. We will be discussing this problem in Section V.

The input for supervised algorithms can be either a structured data type or an unstructured data type, such as free text. The data needs to be transformed from textual to numerical data when the input is in free text format. This transformation is normally done through the use of the bag-of-words model [34]. In this model, every word in the document collection that satisfies certain criteria (e.g. a minimum frequency threshold) will first be counted on occurrences. A new text is then represented by the frequency of occurrences. If the data contains a rich amount of unique words, the resulting occurrence matrix will be sparse.

However, some words occur many times even though they do not add to the context of a sentence, while other words which make a sentence unique occur rarely. Therefore, after the occurrence matrix is obtained, another measure will be done, called term weighting. The universal term weighting method is the 'term frequency inverse document frequency' (*tf-idf*) [23]. This weighting of words will divide the frequency of a term in a document by the overall frequency of the word in all documents. Term frequency inverse document frequency is calculated through Equation 1:

$$\text{tf-idf} = tf * idf \tag{1}$$

$$tf_{t,d} = \begin{cases} 1 + log_{10} * tc_{t,d}, & \text{if } tc_{t,d} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$idf_t = log_{10}(\frac{N}{df_t}) \tag{3}$$

Here, tf is the term frequency, idf is the inverse document frequency, d a document in document collection D, $tc_{t,d}$ is the term count (the number of times that term t occurs in document d), N is the number of documents, and df_t is the document frequency (the number of documents that term t occurs in.

A more sophisticated model is required to leverage semantic relationships between words. The representation of textual data can be represented by means of embeddings [35], reducing the dimensionality of the numerical representation of words. The algorithm to obtain the numerical representation is called *word2vec*, where the created vector is the new representation of the word. The distance between two vectors can be calculated to represent the similarity between two words. One problem may arise when the vector representation for a word must be used to create one vector representation for a sentence. We will propose our solution to this problem in Section IV-D2.

Because the dataset had numerous different labels, the classification problem can be specified as a multi-class classification problem. This has been widely studied in the world of classification problems [1] [24]. All classifiers initially assign one class to the input. The interesting part of our research is that we would still like the user to choose which class the input belongs to. The classifier will only recommend classes to the accountant. Therefore, we were free to allow the classifier to suggest more than one class for one input.

B. HYPERPARAMETERS TO TUNE

With the background of classification algorithms in mind, one should think about the number of different (hyper)parameters that play an important role in classification. First of all, several algorithms have been introduced which are useful in text-mining problems [8] [26] [1]. Most algorithms have been proven to predict accurately with different types of data. Therefore, we had to establish which algorithm had the most potential for the specific problem of multi-class classification of bank transactions.

Secondly, the number of training instances might have a large effect on the actual accuracy. A model trained with too many instances could be prone to overfitting. Overfitting is the issue that the model has perfected the predictions on the training instances and does not generalise well with unseen instances. On the other hand, too few instances would not reach the full potential of an algorithm.

Thirdly, feature selection and feature filtering might create a positive effect on accuracy. The number of features could explode when every unique word in a free text is a feature. In contrast, Too few features could result in low accuracy.

C. CLASSICAL MACHINE LEARNING

Based on previous studies, we decided to use naive Bayes (*NB*), Support Vector Machine (*SVM*), random forest (*RF*), and xgboost.

1) Naive Bayes

Naive Bayes is mostly seen as an algorithm that is easy to understand, and works notably well with classification problems [2]. While the algorithm is also suitable for multi-class classification problems, it is best known for its wide usage in binary classification problems, for example, e-mail spam-filtering [43]. We considered this algorithm as our first baseline.

The calculation of naive Bayes is done through probabilities. The input consists of the Document series D, where $d \in D$. The probability that class c belongs to document d is calculated through Equation 4.

$$P(c|d) = \frac{P(d|c) * P(c)}{P(d)} \tag{4}$$

The class with the highest probability for one document will then be chosen as its label. For $P(d|c)$, the naive assumption can be made that it can be calculated through the product of each term t based on Bayes' theorem [5]. In short, the probability of a document's term dictionary given a specific class is calculated as the product of the probabilities of each term given a class. This is done through Equation 5.

$$P(d|c) = P(t_1|c) * P(t_2|c) * ... * P(t_k|c) \tag{5}$$

Here, t_1 is the first term of one input document, with k being the number of terms in a document, c is the given class, and d is the document. The

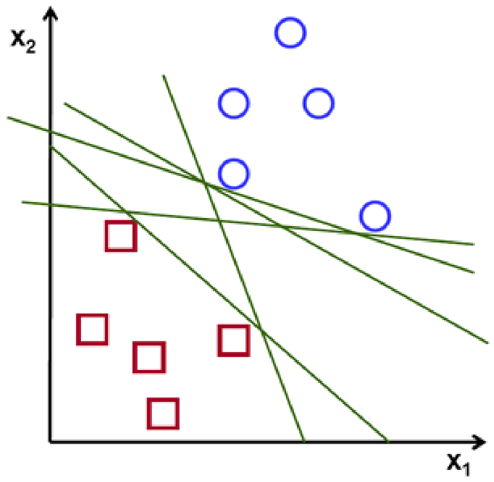

Figure 1. A collection of different separating hyperplanes in a 2-dimensional vector space [16].

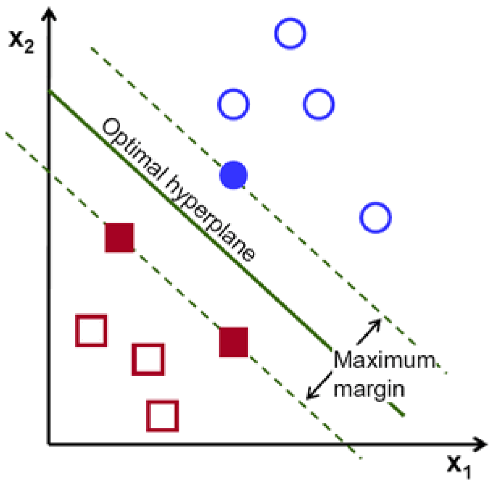

Figure 2. The optimal hyperplane found for a 2-dimensional vector space [16].

probability of class c belonging to the input term t can be calculated through Equation 6.

$$P(t|c) = \frac{Tr_{ct} + 1}{(\sum_{t' \in V} Tr_{ct'}) + |V|} \tag{6}$$

Here, Tr_{ct} is the number of occurrences of t in training inputs from class c, $|V|$ is the number of different terms and $\sum_{t' \in V} Tr_{ct'}$ is the total number of term occurrences in training inputs from class c.

2) Support vector machine

The Support Vector Machine algorithm has proven to be extremely useful in text-mining classification problems [52]. SVM tries to find a hyperplane in an n-dimensional vector space that will divide every data point into a distinct class. Here, n is the number of unique classes. The algorithm is best explained with a two-dimensional vector space since this algorithm will scale no matter how large the dimension size is. For a more technical description of the SVM algorithm, see Hearst et al. (1998) [21].

Figure 1 illustrates classification by any hyperplane, while Figure 2 illustrates classification by the optimal hyperplane. Every data point on the figure is an input, with the squares and circles representing the different classes. By continually 'drawing' different hyperplanes and calculating the mean distance to all data points from one class, the algorithm will eventually find the optimal hyperplane for one class. If the mean distance for both classes is at an optimum, then the best hyperplane has been found. The most important data points to decide the optimal hyperplane (e.g. the two square data points and one round data point on the dotted lines in Figure 2) are called the support vectors. If a new data point will be used as input where the algorithm does not know which class it belongs to, it will be placed in the vector space, and depending on where it is placed, the particular class will be assigned to it.

SVM makes use of so-called 'kernel' functions. Kernel functions are a set of mathematical functions to manipulate the data. The kernel functions are used to calculate the inner product between two points in a standard feature space. This means that the SVM can use the data in a higher dimension of the feature space, no matter the dimension surface of the data. Well-known kernels are the linear kernel, Gaussian kernel, and the sigmoid kernel.

3) Random forest

Random forest is another well-known machine learning model [7]. random forest is an ensemble method, meaning it consists of different smaller models which are combined into one large model. As the name 'forest' suggests, the smaller models are all decision trees. Every decision tree uses features to split the possible classes into different leaves from the tree. The feature splitting is done through random initialisation and the quality of a particular split is then evaluated by a metric such as entropy [25] (Equation 7, with D the entropy, K the number of classes with k as a particular class, m the chosen split region (one node in the decision tree), and $\hat{p}$ the proportion of training observations in the mth region from the kth class). All models are uncorrelated from each other because the trees are made from random subsets of features. All models are trained independently and running a test set on all of these models will yield one most predicted class, which will be the class random forest predicts. This method is known as bagging, or bootstrap aggregation [6]. The combination of both bagging and random feature decision trees makes random forest less prone to overfitting or variance.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk}) \tag{7}$$

4) Xgboost

Similar to random forest, xgboost (Extreme Gradient Boosting) is also an ensemble method [9]. Xgboost consists of several decision trees. However, the greatest difference with random forest is that xgboost uses boosting instead of bagging. Boosting is a technique where instead of creating decision trees in parallel, the decision trees are built on top of each other, learning from errors of earlier created models. Gradient boosting uses a gradient descent algorithm over an objective function to correct the errors present in its predecessor. Gradient descent algorithms tend to iteratively find a local optimum with the help of the objective function. Xgboost also assigns weights to all of the individual features. In each training round, the weights of the features with a wrong prediction are increased and used in the next decision tree. In the end, the weighted sum of all decision tree predictions, over the course where each decision tree learns from its predecessor, is the final xgboost algorithm.

D. WORD EMBEDDINGS

More recent classification algorithms are deep learning algorithms [36]. Such algorithms use complex, layered models to ultimately categorise the input into the correct output. These layered models, also known as neural networks, usually transform the data in several ways. Deep learning algorithms can be applied to classification algorithms with different techniques. One such technique is the end-to-end model, such as Bidirectional Encoder Representations from Transformers (*BERT*) [13]. However, most end-to-end models need an extensive amount of train data.

Another technique is to apply deep learning for embedding. This would enrich the pre-processing of the input with a semantic relationship between words. The classification part remains the same as with classical machine learning algorithms. Because this application does not require an extensive amount of train data, we decided to not only use tf-idf pre-processing, but also a word embedding algorithm for pre-processing. This is done with a method called word2vec [35].

As the name word2vec suggests, it transforms a word into a vector. The dimension of the embedding space is a parameter that can be set freely. Furthermore, the vectors of all the words are an embedding of words [19]. This means that every word correlates with the other words. During the training of the model, the model tries to find words that exist in similar contexts. If two words of similar context can be linked to each other, then the similarity of these words is high. With such a pre-processed model, the input of the classification algorithm might have some correlation between words, instead of the more simplistic tf-idf pre-processing. Because word2vec creates relationships between words, we expected that word2vec might further increase the accuracy of our model.

E. GENERAL PIPELINE

The algorithms explained in the previous sections need to be combined to have a fully functional model predicting ledger accounts from bank transaction descriptions. There is a pre-processing part and a learning part. We have two different pipelines since we used different algorithms for both parts. Either:

a) Classical machine learning:
   1) Vectorizing (bag-of-words)
   2) Transforming (tf-idf)
   3) Classifying (naive Bayes, SVM, random forest, or xgboost)
b) Combination embedding and classical machine learning:
   1) Vectorizing and transforming (word2vec)
   2) Classifying (SVM)

IV. EXPERIMENTS

This section covers all the conducted experiments to analyse and improve the accuracy of predicting the ledger account of the bank transactions. Firstly, four different algorithms had been run on the provided dataset, naive Bayes, support vector machine, random forest, and xgboost. Secondly, feature selection was used to experiment with different amounts of features, because free text as input might contain many unique words, which determines the number of features. Thirdly, the number of instances for the training dataset was changed on a fixed test set. This experiment was run on multiple datasets for verification. Finally, two different pre-processing methods were used.

The models had been compared on the accuracy they obtain. In addition to top-1 accuracy, the top-n accuracy was also covered. Top-n accuracy is defined as the accuracy over n-classes. If one of the top-n classes with the highest probability is the correct class, then the suggestion of the model is correct. This design was chosen because of the large number of unique classes

Bank transaction component	definition
Component name	bank transaction
Date	01-01-1700
Counter account	
Counter account name	AnYoi
Amount	10
Reference	REFRNR. AAA111
Description	BEA NR:619FGT 01.01.16/10.38 SHELL NIJKERK
Alternative Type	receipt
Ledger account Id	11111111-1111-1111-1111-111111111111
Ledger account name	revenue low
Ledger account number	1000
tax percentage	
tax	

Table 1. A fictional example of one bank transaction in the dataset.

and because we only have to suggest ledger accounts. Before any experiments were run, a thorough understanding of the datasets was required.

A. DATASETS

The data was collected and provided by AFAS Software. AFAS Software is a Dutch software company (Independent Software Vendor, *ISV*) that delivers enterprise resource planning (ERP) systems for other companies and businesses [46]. Their current customers use a system which is called 'Profit'. Profit is widely used in the Netherlands with a large installed base. At the same time, AFAS is rebuilding its Profit ERP application on a brand new application platform called 'Focus'. A dataset can be obtained from both systems, with Profit holding an extensive amount of data while Focus will contain more data in the near future. The target for our research is Focus, even though we experimented with the Profit dataset to uncover if using more training instances had a positive effect on accuracy of the model.

Both datasets have one transaction per line. Every column is a different attribute of the transaction. An example of a bank transaction can be seen in Table 1. Moreover, all transactions are already classified with two columns for the ledger account description and number. Our main input column will be the transaction description. Other columns involve the tax category, amount of the transaction, name of the organisation, counter accounts, and more. Such information from a bank transaction can also be used as input for the machine learning model. However, these were not considered due to the scope of this thesis, as we focused on information extraction from free text.

Our main output column is a concatenation of the ledger account name and ledger account number, which is the case for every model. This will create more unique classes, as every customer can create their own ledger account names. However, if the model predicts only the ledger account number, the predicted number might not reflect the same ledger account for one customer compared to other customers. The datasets also contain ledger account numbers without a description Therefore, the combination of the two was the most useful output column. Since all data is customer data from AFAS, the data is strictly forbidden to be used outside of AFAS and follows the rules of the EU General Data Protection Regulation [18].

The Profit dataset consists of 1.383.319 transactions. Such an extensive amount of data is a good indication of how much an algorithm can learn from train data. Furthermore, the algorithms can be tested to see what the computational cost is for these inputs. The Focus datasets consist of respectively 10.698 (Focus A) and 21.062 (Focus B) instances. At the start of the research, the Focus software had only 10.698 instances of transactions. As AFAS obtains more customers, the transactions will gradually increase as well. Some experiments were run with Focus A before AFAS was able to provide Focus B.

Every unique ledger account belongs to certain higher-level categories. For instance, revenue systems, revenue components, and revenue consultancy all belong to the same category, revenues. An example of the hierarchy levels can be seen in Figure 3. The ledger accounts are part of a hierarchical tree. The models predicting the outcome will predict the lowest leaf value of the tree. This design is chosen as there belongs a 1:1 mapping between transactions and the leaves in Figure 3, which are the ledger accounts. In all current datasets, there is no direct mapping between the parent nodes in Figure 3 and the bank transactions. The parent nodes can instead be derived from the leaves.

A challenge in classification with this dataset will be the skewedness of the outcome distribution (see Figure 4 for the twenty most frequent classes). There are 447 unique classes in Focus B, with 94 of these classes appearing only once. This means that 21% of all unique classes only appear once. Some classes, on the other hand, appear more frequently than other classes. Therefore, the classes are non-uniformly distributed over the inputs. It should be noted that accuracy is the central metric in the experiments. The production model should suggest more than one ledger account since the number of unique classes is high. Therefore, top-n accuracy is more

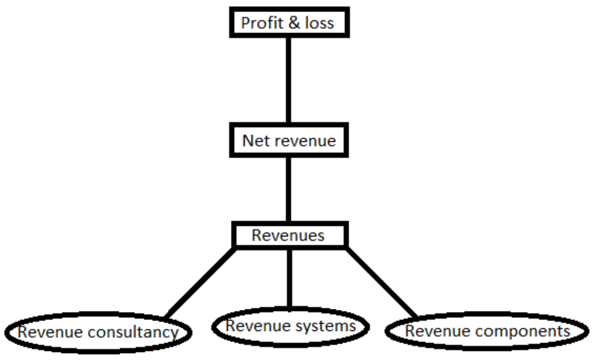

Figure 3. An example of the different hierarchy levels for ledger accounts. The leaves represent different ledger accounts, while the nodes represent different sub-categories.

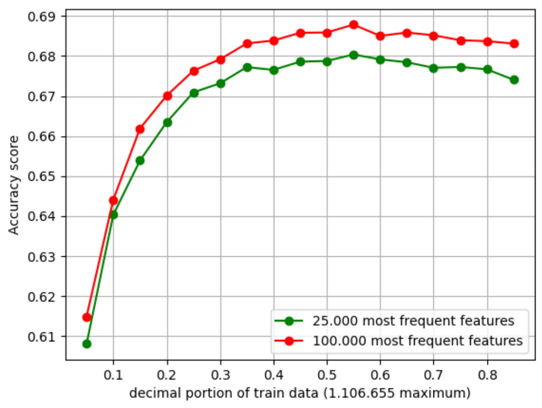

Figure 5. Accuracy of SVM on fixed Profit test set when increasing the number of Profit training instances with 25.000 and 100.000 most frequent features respectively.

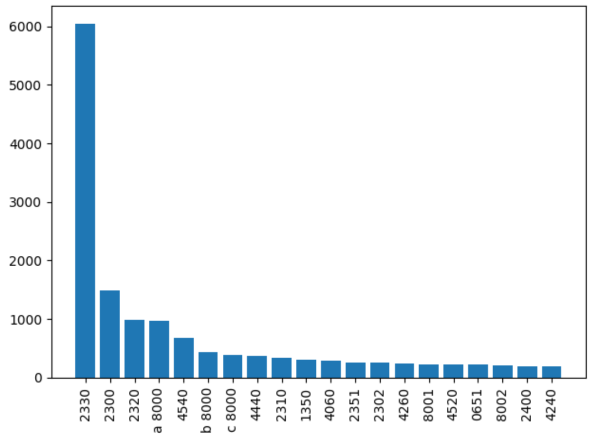

Figure 4. The 20 most frequent classes for concatenated ledger account name and number. The names have been filtered. For reference, the 94 least frequent classes all appear only once.

useful than other metrics.

B. EXPERIMENTAL SETUP AND RESULTS ON PROFIT DATASET

The most logical experiment to take for the vast amount of data of the Profit dataset was to investigate to what extent the size of the dataset could improve the accuracy of our test set. Therefore, we decided to make a fixed subset for testing and continuously raised the amount of training data. The Profit dataset would be the baseline for the number of available instances that could have a positive effect on accuracy.

The data was read and pre-processed through a tf-idf transformer. However, it was decided to only use a fixed amount of most frequent features, as this parameter choice was already experimented with in Section IV-C. There are a lot more features, so it was chosen to use two different amounts of the most frequent features. These are respectively 25.000 and 100.000 features. Furthermore, the algorithm that was used for this experiment is the support vector machine algorithm with a linear kernel. Cross-validation was left out due to the massive amount of data available, which means that the computational cost of the algorithm was already high. The choice for this specific algorithm had been experimented with in Section IV-C1.

The total train set consisted of 1.106.655 instances because the dataset was first split into an 80% train set and 20% test set. The initial experiment was run with a test set consisting of 276.664 instances. However, during the increase in the number of training instances, the model ran into memory issues with the test set. Therefore, the fixed subset for testing had been further cut off to 30 percent of the initial test set. This resulted in a fixed test set of 82.999 instances. Results with both fixed amounts of features can be seen in Figure 5. The accuracy reached an optimum of around 600.000 instances for both the model with 25.000 features and the model with 100.000 features. The difference in accuracy between these models was minimal, while the training time for 25.000 features took around 25% less time compared to 100.000 features.

C. EXPERIMENTS ON FOCUS A

A range of experiments had been conducted on the Focus dataset. This section will cover both the smaller (A) and the larger (B) Focus datasets. Each subsection covers one experiment and all discussions of the experiments can be found in Section V.

Algorithm	Accuracy	Recall	Precision	F1
naive Bayes 1 column	0.4759	0.0855	0.1186	0.0861
naive Bayes 2 columns	0.4513	0.0806	0.1100	0.0800
linear SVM 1 column	0.6183	0.2632	0.3434	0.2759
linear SVM 2 columns	0.6183	0.2676	0.3435	0.2825
xgboost 1 column	0.5747	0.2261	0.2538	0.2276
random forest 1 column	0.6405	0.3296	0.3865	0.3382

Table 2. Results of different algorithms run on Focus A.

1) Experimental setup and results with different algorithms

First of all, it was important to use the same setup with different algorithms, to get a better understanding of which algorithm had the most potential for this specific data. A total of four different algorithms had been chosen, with naive Bayes being the baseline. The other algorithms include SVM, random forest, and xgboost.

The hyperparameters of each algorithm were set to a constant value, as we first wanted to experiment with the potential of different algorithms. Experimenting with the hyperparameters would increase the computational time significantly. Focus A was used, in which for every algorithm a different train and test split were chosen, with 80% train and 20% test. Because the splitting is random, the numbers could change a little for every run. Cross-validation was not yet considered, because the precision of the accuracy would not affect whether an algorithm has potential for further experiments. The input data was pre-processed through a tf-idf transformer. The main input column was the description of the transaction. However, we also concatenated a transaction reference (e.g. REFRNR. AAA111, see Table 1) to the description column to see if that would affect the training. Results can be seen in Table 2.

As can be seen in Table 2, both SVM and random forest are the most promising algorithms. The F1 scores reached around 30%. The ledger accounts were counted to see whether or not the algorithms only predict the most common ledger accounts. The most common ledger account appeared in 8% of the total data. Therefore, the accuracy shows that most algorithms have learned from the input data and not just suggest the most frequent ledger account.

2) Different folds cross-validation

After experimenting with different algorithms, the experiment to alter the number of folds in cross-validation was carried out. For this experiment, a linear SVM was used in combination with increasing the cross-validation folds. By using cross-validation, a more reliable estimate of train and test set errors can be obtained. Cross-validation is a technique to divide the data into train and test sets. The results can be seen in Figure 6. Increasing the number of folds means the train set has more instances while the test set has fewer instances. Increasing the number of training instances results in higher accuracy, while a small decrease in accuracy can be seen from the train set. Both results were expected, as increasing the number of instances will also increase the number of unique classes in the train set and more classes are known and predictable in the model for the test set.

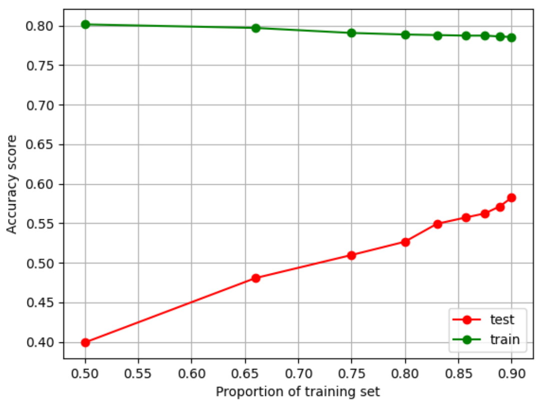

Figure 6. Accuracy of SVM with different cross-validation folds on Focus A. More folds results in a larger training set.

3) Cross-validation for accuracy verification

Validation methods are normally used to ensure the precision of the results a model predicts. The accuracies obtained in Table 2 were not precise due to the randomness of splitting the dataset into one train and test set without cross-validation.

One smaller experiment had been run to verify the accuracies obtained in Table 2. 5-fold cross-validation was used on SVM, naive Bayes, and random forest. Xgboost did not achieve the accuracy SVM and random forest obtained when we compare the accuracies in Table 2 and was therefore dropped from this experiment. 5-fold cross-validation means the sets were divided into an 80% train and 20% test set. The same pre-processing had been done as with Section IV-C1 with Focus A. Results can be seen in Table 3. Here, the accuracy of SVM was a bit higher compared to random forest, although the accuracy obtained after cross-validation was lower compared to the accuracy obtained with the smallest Profit train data (see Figure 5 for comparison). Following experiments were carried out with SVM

Algorithm	cross-validated accuracy
naive Bayes	0.3926±
SVM	0.5243±
random forest	0.5209±

Table 3. The accuracies reached with different algorithms using 5-fold cross-validation with Focus A.

because it obtained the highest baseline accuracy.

4) Feature selection

The current feature selection is based on the words that occur in the description column. Everything that consists of a combination of letters, digits, and signs is seen as a word. Currently, every feature consists of one word, a one-gram. Another experiment had been run with two-grams, three-grams, and four-grams as features. The initial results, however, were not promising enough for further exploration.

A small analysis of the data of how many features a dataset has shown Focus A with 11.301 features from 10.698 instances, Focus B with 15.056 features from 21.062 instances, and Profit dataset with 1.348.707 features from 1.383.319 instances. This means that an increase in the number of instances will also increase the number of features. Since such an extensive amount of features will not be useful, an experiment was run to see if there is an optimum amount of features.

The setup for feature selection consists of Focus A, an SVM with linear kernel, and 10-fold cross-validation. The number of features was changed with each run, where the results can be seen in Figure 7. The features are chosen with the highest count first. This means if there are 1.000 features chosen, the 1.000 most frequent features will be used during training. There seems to be an optimum at 3.000 features, which in this case means around the 30% mark. Some experiments used a fixed amount of most frequent features (e.g. 3000) while other experiments consisted of a fixed percentage of most frequent features (e.g. 30%).

D. EXPERIMENTS ON FOCUS B

An updated version of the Focus dataset was obtained, which had more than twice the amount of instances compared to the first dataset. As we have seen in Section IV-B, increasing the amount of training data resulted in higher accuracy. Therefore, a comparison between Focus A and B, the larger Focus dataset, should result in higher accuracy with Focus B. Furthermore, we introduced an embedding method for pre-processing the data, such that the feature vectors

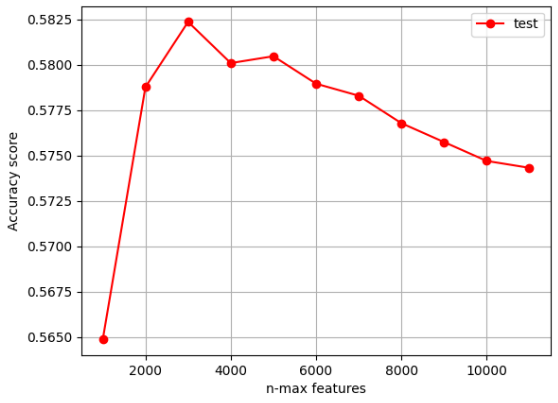

Figure 7. Accuracy of 10-fold SVM with different amount of features from Focus A.

are not word counts but rather complex vectors which were trained to embed similar words close to each other. Together with this experiment, the combined accuracy of top-n classes was given. The accountant has the choice between a top-n most likely classes, and therefore our accuracy should be given for not only the most likely class but also for the top-n most likely classes.

1) Training size comparison between Focus datasets A and B

First of all, a comparison between datasets A and B might give more information on how much growth there was to be gained with more Focus training data. For this experiment, the data of both datasets A and B was tf-idf transformed. To see if there was an increase in accuracy for a test set between two datasets, the test subset should contain almost the same amount of instances for both datasets. a 20% test subset had been chosen for Focus A, while a 10% test subset had been chosen for Focus B. The training dataset gradually increased for both datasets, until the complete train subset was used for both datasets. An SVM was used with a linear kernel, and 10-fold cross-validation was used on the training data. 35% most frequent features were selected for both datasets. The results can be seen in Figure 8. The accuracy scores for Focus A range between 45% and 64%, while the accuracy scores for Focus B range between 57% and 73%.

2) word2vec as a pre-processing step

In this experiment, we compared the traditional tf-idf weighted bag of words representation used for our transaction description column input with a trained word2vec model. Since the model needs to do suggestions for accountants, we also intro-

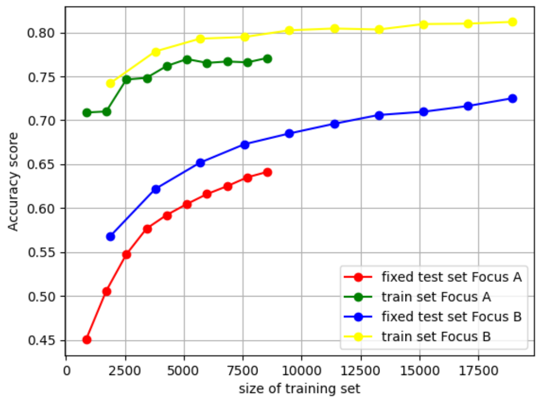

Figure 8. Accuracy of 10-fold SVM on datasets A and B with an increasing amount of train data.

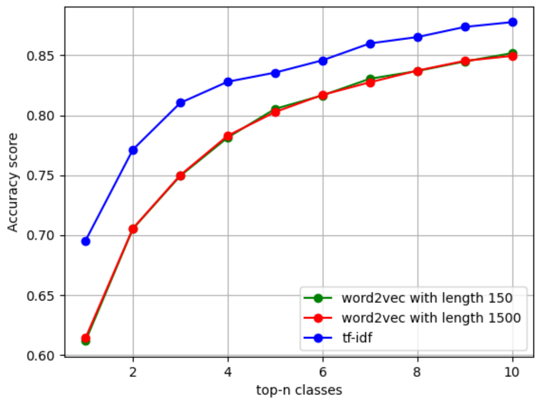

Figure 9. A comparison between three different pre-processed data for the combined accuracy reached with top-n classes using 10-fold cross validation SVM on Focus B.

top-n accuracy	tf-idf	word2vec 150	word2vec 1500
1	0.695	0.611	0.614
2	0.771	0.705	0.706
3	0.81	0.749	0.75
4	0.828	0.781	0.783
5	0.836	0.805	0.803
6	0.846	0.817	0.817
7	0.86	0.83	0.827
8	0.865	0.837	0.837
9	0.874	0.845	0.845
10	0.878	0.852	0.849
standard deviation	0.056	0.076	0.074

Table 4. The combined accuracy values obtained when looking at the top-n predicted classes with different pre-processed data on Focus B.

duced the top-n classes' combined accuracy.

The word2vec model must first be trained to obtain the most accurate word vectors. Instead of using all unique words in the description as with tf-idf, we filtered any non-alphabetic words from the feature list. We did not know the meaning behind all non-alphabetic words, for which we assumed that it could interfere with the embedding of these words and alphabetic words. A revised version could include some of these non-alphabetic words, although this is out of the scope of this thesis. All words were lower-cased to ensure different occurrences of the same word are all the same after the filter process.

A trained word2vec model can give a probability for two words how much they are related to each other. The model had been trained with a vector length of 150 as well as a vector length of 1500. The tf-idf transformer created sparse vectors with a length as long as all the features. This was not feasible for word2vec, since the word2vec model itself must also be trained and the process would take too long if the length increases. 10 epochs were chosen such that the model does not overfit, but would benefit from more training time. The word2vec model was then saved for later use during the actual training of the SVM.

The input in the linear SVM model was a vector of either 150 numbers or 1500 numbers for each transaction. For the word2vec model, the vector for the complete sentence was created through a summation of the vectors for each word in the description. If a word does not appear in the word2vec model, then a vector of zero's was chosen, such that it does not impact the trained model. As a final step, the summed vector was divided by the number of words that had an existing vector. Due to this computation, a $sentence2vec$ was created (Equation 8), which was exactly what was needed for the SVM. Here, k is the amount of non-zero word vectors.

$$sentence2vec = \frac{\sum_{i=1}^{k} word2vec_i}{k} \tag{8}$$

During the actual training of the SVM model, 10-fold cross-validation was used for every increasing top-n class accuracy. The 10-fold cross-validation was changed in each iteration, which accounts for a small bit of randomness during the experiment. The results can be seen in Figure 9. These results also contained a trained linear SVM model where the inputs consisted of a tf-idf transformed vector. This tf-idf transformed vector used all features. For reference, the actual results of the experiment can be found in Table 4. The tf-idf transformed vectors obtained higher accuracy compared to both word2vec vectors.

V. DISCUSSION

We will discuss the experiments individually, because some results from earlier experiments were used in later experiments. Each subsection covers the discussion of one experiment, carried out in chronological order. Furthermore, some additional topics are covered in subsection V-B.

A. EXPERIMENTS DISCUSSION

Since every model makes suggestions for an accountant and is not decisive, the incorrect suggestions are less of an issue. Furthermore, as established in Table 2, some algorithms can not only predict the most frequent classes but use the input for a more accurate suggestion. The final experiment shows a top-3 accuracy of +80% (Figure 9). While such accuracy is desirable, there is enough room for even more improvement. Some of these will be discussed in Section VI.

From all the experiments that were done, classes with only a few transactions or just one transaction were difficult to predict. The absence of many instances for a unique class was the general issue throughout the experiments. Classification algorithms generally cannot suggest a class that does not exist in their training set. To partially solve the growing issue when accountants introduce more and more ledger accounts, is to let the model train during new inputs. This would suggest an online learning environment for the model and as such real-time analysis of the correct labels. When an accountant labels the new transaction, the transaction itself is sent to the model and added to the training dataset. The trigger to send the labelled transaction would be for every transaction that is classified manually by the accountant. We could also let the model learn when an accountant selects one of the suggestions by the model. However, the model is more prone to overfitting if the model learns from correct suggestions.

1) Algorithm choice

Many experiments were run with an SVM. Table 2 shows that both SVM and random forest have some potential for this specific problem. However, one decision tree with a small depth will not be able to accurately distinguish more than 447 classes. Even though random forest is an ensemble that consists of different decision trees, an increase in depth would be beneficial. If the depth increases, the learning time for a decision tree algorithm will also increase. Furthermore, the memory needed for random forest increases with the number of instances and the number of trees used. Therefore, due to these scalability issues, SVM was chosen to be the main algorithm for most of the experiments.

2) Feature selection

The description column has many different words, some might contain abbreviations, some might have reference numbers, and other non-dutch or non-English words. As the tf-idf transformer considers every unique word from the input a unique value, our dictionary can become very large. The results obtained in Figure 7 thus show that unique values, such as a combination of digits and letters from a specific company, are not useful for training a classifier. There has to be some caution, as the amount of features increases with the number of instances. A fixed amount of most frequent features might not guarantee the best results but was used when large amounts of training instances are available. A fixed percentage was also considered for experiments with Focus B. The optimum found in Figure 7 was 3.000 most frequent features, which translates to around 30%.

3) Number of instances for training

While the amount of features is important for training time, the number of instances could have a larger impact on the accuracy of the model. As can be seen in Figure 6, an increase in the number of cross-validation folds, which leads to an increase in train data, will net a better test score. To further prove this, the experiment carried out in Section IV-B consists of continually increasing the number of training instances with a fixed subset of test instances. The biggest implication that can be seen in Figure 5, is an optimum for the number of train instances in the Profit data. Since the accuracy will not improve sufficiently after around 600.000 instances, there is no need for even more training instances.

When the test scores are compared in Figure 8, the model trained on Focus B (the larger dataset) scores higher accuracies for both train and test set compared to the model trained on Focus A (the smaller dataset), even when the size of both training sets is the same. This could be because the training instances in Focus B might have richer information in the transaction description column compared to Focus A. It is also worth noting that, even though 10-fold cross-validation is used, the random splitting into train and test sets might have slightly different results each time the experiment is run. This impacts the smaller subsets more than the larger subsets, which might also impact the difference in test accuracy for datasets A and B.

Furthermore, the curve of the test set for Focus B looks to be flattened, while the curve of the test set for Focus A is still upwards. The flattened curve could suggest that the model has an improved generalisation with Focus B, or the curve suggests that the SVM model would need a lot more instances for a small increase in accuracy.

A comparison between the curves in Figure 5 is insightful for the number of features and the training time. There are more than 1.000.000 features in the Profit dataset if all the data is used. The suggested 30% in Section V-A2 yields more than 300.000 features. The first run with such an amount of features resulted in memory and computational errors. To still see whether an enormous amount of features was useful, two experiments were run with 25.000 features and 100.000 features. As can be suggested from Figure 5, the increase in accuracy is minimal compared to the extra computational cost. This is in contradiction with the suggested 30%, as 25.000 features are less than 2.5% of the total features. Therefore, a maximum amount of features might be more useful than a fixed percentage of the most frequent features when a large amount of instances is available.

4) Word2vec pre-processing

While tf-idf is widely used in text classification problems, it does not take into account semantic similarity. Words within the same context, abbreviations of words, and subject-equal words are not considered in a tf-idf transformer. Therefore, an embedding might be of use, to calculate distances between words and create a vector for each word. However, stated in Section IV-D2, the dimension of the vector for one word must be cut down significantly before it can learn in sufficient computational time.

When the results in Figure 9 are compared, the accuracy of a simple tf-idf transformer is even higher than the word2vec input. This could very well be associated with the dimensionality reduction compared to the tf-idf representation. If the dimensionality of the word2vec representation would be closer to the dimensionality of the tf-idf representation (15.056), then the result might become different. The differences between both word2vec results in Figure 9 are insignificant, although the training time was a bit longer for the word2vec model with 1500 size compared to the word2vec model with a length of 150.

Another possible explanation for the lower accuracy of the word2vec model might be due to the context of the descriptions and the ledger accounts. The relevance of semantics between words might not be useful in this classification, as accountants will probably not create ledger account labels based on the sentences in the transaction description, such as ‘Shell Nijkerk’ in Table 1. Such semantics might be more useful in combining descriptions with open invoices, as an accountant can directly issue an invoice to ‘Shell Nijkerk’, whereas categorising into the ledger account ‘revenue low’ has less context.

B. DISCUSSION OF THEORETICAL CONCEPTS

The discussion in Section V-A covers all the discussions about the results of the experiments. However, there are some other concepts for this specific problem that needs to be discussed. The models return the labels of the ledger accounts they predict, as well as the number that corresponds to the ledger account. While this is a normal design for most models, a lot of labels could be company-specific. In the software AFAS created, every accountant can re-assign labels for most of the ledger accounts. Furthermore, an accountant can create even more for whatever purpose. The current model is a global model that learns from all bank transactions from every company. However, a new problem arises that some accountants might obtain a ledger account name or number from the model that they do not know. Furthermore, some labels might contain sensitive information, therefore breaching privacy laws.

To solve these issues, one might introduce company-specific models. This would indicate that one model cannot learn data from other companies, which could drastically lower the accuracy of most models. To avoid having a different model for each company, the model could also be minimised to suggest only the general ledger accounts that AFAS provides to every new customer of them. However, a more complete solution would be to build a filtering machine, that would filter any suggested ledger account for a company that should not be allowed to see the suggested ledger account. The model does not need to be changed with this solution and is as such the recommended solution.

Another possible workaround for any privacy-related issues would be to suggest a certain hierarchy level instead of the ledger account itself. All ledger accounts are sub-categories of higher-level categories, where the whole ledger tree can have multiple levels of hierarchy. This would mean that the model would suggest ‘Revenues’ or ‘Net revenues’ instead of ‘Revenue consultancy’ (Figure 3). While this would be sufficient, the accountant still has to manually decide which of the ledger accounts the new transaction would belong under ‘Revenues’. As explained in Section IV-A, the higher-level category can be derived from the predicted ledger account. This would mean that the predicted higher-level category is only based on the predicted ledger account. Explicitly predicting the higher-level category would result in a multi-dimensional classification problem, where the newly trained model not only predicts a ledger account, but also the higher-level categories.

VI. FUTURE RESEARCH

While many experiments had been conducted, there is still much left to experiment and analyse. There are many more possible approaches to experiment with-not only the data pre-processing part, but the model itself as well as the output that is given by the model.

A. INPUT CHOICE

Even though the accuracy reached with the current description input is a positive result, different inputs might lead to more convincing predictions. For example, the counter account or the date might contain more information that is useful for classification. Some models have multiple different features as input. While the data in Table 2 already contains the results of experiments done using multiple columns, the columns are modified such that only one kind of feature is used. There could exist relationships between other inputs and the ledger accounts that are not found from the transaction description column alone (e.g. either classify the counter account or use the counter account as another feature). Before the multi-feature option is considered, one could also use the ‘amount’ column and divide it into segments, such that the smaller amounts are categorised into the same segment. These segments could just be concatenated to the description column, which in turn would make the description column a bit richer in text.

Even though the experiment with different n-grams (Section IV-C4) did not have promising initial results, another experiment could be run where the n-grams are combined. This would mean that not only one-grams exist as a feature, but also the most frequent n-grams. The number of features will drastically increase, so it would be interesting to see whether such an experiment has positive results on the accuracy.

B. ALGORITHM CHOICE

While random forest and especially SVM already have promising results, many more algorithms for text classification are created over the years. In the classical machine learning department, one might suggest a k-nearest neighbour algorithm [10]. This algorithm uses distance calculation (e.g. Euclidean distance [12]) to calculate the distances of its k-nearest neighbours and classify the new data point depending on the classes of those neighbours. This algorithm might work notably well in combination with word2vec because the word2vec model is trying to create vectors where words with similar meanings tend to be close to each other in the dimensional space.

More algorithms are available in the area of deep learning which might be useful for the text classification problem. A neural network with multiple layers would then be the most evident algorithm to use. Since the output is multi-class, the final layer of the neural network should have one neuron for each class. The neural networks range from the classic neural networks, such as multilayer perceptrons [17] (multiple feed-forward layers with activation functions, each layer has weights defined before each epoch) or the backpropagation network [22] (the weights are trained on the amount of error from the last epoch) to more advanced neural networks, such as BERT [13], roBERTa (Robustly Optimized BERT pretraining Approach) [31], DistilBERT [44], or ALBERT (A Lite BERT) [28].

The biggest challenge for such an algorithm would be the amount of data. Normally, a neural network needs massive amounts of training data to be of any use. The amount of data currently available from Focus would likely not be enough to be able to successfully train a neural network. Furthermore, the output neurons needed for the current Focus dataset would be 447, which will increase with more instances. The dataset obtained from Profit contains enough instances for such a task. However, the datasets cannot be combined, as the ledger accounts in Profit could be defined differently compared to Focus. As AFAS obtains more and more data from Focus, a neural network might be very useful in the future.

Another option would be to experiment with algorithms that are already used. Most parameters of the algorithms had been set to basic constants, to reduce the number of possible variables. The fine-tuning of algorithms could have high computational costs for a small increase in accuracy. Therefore, the choices with the largest impact were more important to experiment with during this thesis. However, a grid search over the algorithm parameters, for example, could be used to obtain slightly higher accuracy. The training time for such a grid search could become lengthy, depending on the number of training instances. Especially when more than one algorithm is analysed.

C. OUTPUT CHOICE

The current output consists of a ledger account label and corresponding number. However, an accountant might want additional information regarding the new transaction. One option would be to include the tax information, whether the transaction belongs to the high or low tax category. Another option would be to also include the higher category to which the ledger account belongs. To include such options, the train and test set should already consider the extra information as another label to predict. This would transform the multi-class classification problem into a multi-label classification problem. In addition, when predicting the higher categories, the multi-label classification problem becomes a multi-dimensional classification problem. The model would predict a class on more than one hierarchy level. These additional classifications could be more difficult to solve.

Another change that could be of use for this problem would be to introduce a second output with only the ledger account number. Although accountants can create new ledger accounts with new labels and numbers, the standard ledger accounts that AFAS provides to its new users are following dutch national labelling [39]. By providing only a ledger account number, the privacy issue discussed in Section V-B would be resolved. It does introduce a new problem, however, as some numbers from ledger accounts from one company might not be related to the same numbers from ledger accounts from another company, therefore predicting a wrong number and introducing false positives to the suggestion (e.g. the model thinks the number is correct based on its training data, although the suggested number is not available in the ledger from a specific company).

D. PRACTICAL RECOMMENDATIONS

All models are based on existing data and local storage as of today. This could be taken as a starting point to turn it into a proof of concept or product. The global model should be stored on a server, such that it will stay live for all customers from AFAS. Furthermore, it should learn from new inputs, so the model should train on new data whenever the customer manually categorises a bank transaction into a ledger account. The best-recommended model based on the experiments is currently a linear SVM model with a tf-idf weighted bag of words representation. Since the number of instances, and thus the amount of features is constantly expanding, a maximum number of most frequent features should be chosen. As of now, 25.000 features will be enough. In addition, the resulted output consisting of a ledger account name and number should first be filtered to see whether or not the customer can see the predicted ledger account, to prevent privacy issues.

VII. CONCLUSIONS

The research question of this thesis was: "Is it possible to predict (with a certain probability) to which ledger account a bank transaction belongs by means of its unstructured data?" The ledger accounts can be predicted with a +80% accuracy and top-3 suggestions using the unstructured data from a bank transaction. The description consists of a small sentence, which means that a model can learn such cases even with an input that has sparse text features. The amount of ledger accounts is substantial, but the model is still able to reach a reasonable accuracy.

The accuracy was reached after experimenting with different algorithms, different amounts of features and instances, and different pre-processing of the datasets. First, we experimented with naive Bayes, support vector machine, random forest, and xgboost. SVM was chosen as the algorithm with the most potential. After that, we continually increased the number of cross-validation folds, which also increases the number of train instances. Furthermore, we continually increased the number of features to find an optimum amount of features.

Additionally, we used a vast amount of Profit train instances to see the potential with massive amounts of data. To validate these results with Focus data, we compared two different Focus datasets. Finally, we compared tf-idf pre-processing with two word2vec models as pre-processing. On top of that, we looked at the top-n classes' accuracy.

We hypothesised that the word2vec model would have a positive effect on accuracy, but instead, it showed a negative effect. Creating a word2vec model with a vector length the same as the tf-idf transformer could prove to be useful, but for now, the tf-idf transformer is better suited for this multi-class classification problem. Furthermore, not all features are needed for an optimum result. The hypothesis that more instances up to a certain maximum would result in a higher accuracy had been proven through the use of different-sized datasets. Once Focus gets large enough, we can experiment with BERT-like algorithms or other transformer models.

The results of this research can be used and incorporated into the Focus software from AFAS, in such a way that accountants can easily pick the right ledger account from a small set of possible options with a probability of around 80%.

References

[1] M. Aly. Survey on multiclass classification methods. Neural Netw, 19(1-9):2, 2005.

[2] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. arXiv preprint cs/0006013, 2000.

[3] B. Arkok and A. M. Zeki. Classification of quranic topics based on imbalanced classification. Indones. J. Electr. Eng. Comput. Sci, page 678, 2021.

[4] C. Bardelli, A. Rondinelli, R. Vecchio, and S. Figini. Automatic electronic invoice classification using machine learning models. Machine Learning and Knowledge Extraction, 2(4):617–629, 2020.

[5] T. Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. Philosophical transactions of the Royal Society of London, pages 370–418, 1763.

[6] L. Breiman. Bagging predictors. Machine learning, 24(2):123–140, 1996.

[7] L. Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[8] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning, pages 161–168, 2006.

[9] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 785–794, 2016.

[10] T. Cover and P. Hart. Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1):21–27, 1967.

[11] J. Cowie and W. Lehnert. Information extraction. Communications of the ACM, 39(1):80–91, 1996.

[12] P.-E. Danielsson. Euclidean distance mapping. Computer Graphics and image processing, 14(3):227–248, 1980.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[14] N. S. Elmitwally. Building a multi-class xgboost model for arabic figurative language. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS), pages 1–4. IEEE, 2020.

[15] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten. Data mining in bioinformatics using WEKA. Bioinformatics, 20(15):2479–2481, 2004.

[16] R. Gandhi. Support vector machine — introduction to machine learning algorithms. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47, june 2018.

[17] M. W. Gardner and S. Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment, 32(14-15):2627–2636, 1998.

[18] GDPR.EU. Eu-general data protection regulation. https://gdpr.eu/data-privacy/, 2022.

[19] Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722, 2014.

[20] D. J. Hand. Principles of data mining. Drug safety, 30(7):621–622, 2007.

[21] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. IEEE Intelligent Systems and their applications, 13(4):18–28, 1998.

[22] R. Hecht-Nielsen. Theory of the backpropagation neural network. In Neural networks for perception, pages 65–93. Elsevier, 1992.

[23] D. Hiemstra. A probabilistic justification for using tf$\times$ idf term weighting in information retrieval. International Journal on Digital Libraries, 3(2):131–139, 2000.

[24] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. IEEE transactions on Neural Networks, 13(2):415–425, 2002.

[25] G. James, D. Witten, T. Hastie, and R. Tibshirani. An introduction to statistical learning, volume 112. Springer, 2013.

[26] A. Khan, B. Baharudin, L. H. Lee, and K. Khan. A review of machine learning algorithms for text-documents classification. Journal of advances in information technology, 1(1):4–20, 2010.

[27] V. Korde and C. N. Mahender. Text classification and classifiers: A survey. International Journal of Artificial Intelligence & Applications, 3(2):85, 2012.

[28] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942, 2019.

[29] R. Leite and P. Brazdil. Active testing strategy to predict the best classification algorithm via sampling and metalearning. In ECAI, pages 309–314, 2010.

[30] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2):539–550, 2008.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[32] Y. Liu, H. H. Zhang, and Y. Wu. Hard or soft classification? large-margin unified machines. Journal of the American Statistical Association, 106(493):166–177, 2011. PMID: 22162896.

[33] H. Luo, X. Pan, Q. Wang, S. Ye, and Y. Qian. Logistic regression and random forest for effective imbalanced classification. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), volume 1, pages 916–917. IEEE, 2019.

[34] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In AAAI-98 workshop on learning for text categorization, volume 752, pages 41–48. Citeseer, 1998.

[35] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. arXiv preprint arXiv:1309.4168, 2013.

[36] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao. Deep learning–based text classification: a comprehensive review. ACM Computing Surveys (CSUR), 54(3):1–40, 2021.

[37] F. Mohr and J. N. van Rijn. Fast and informative model selection using learning curve cross-validation. arXiv preprint arXiv:2111.13914, 2021.

[38] J. Ojala. Machine learning in automating bank statement postings. Helsinki Metropolia University of Applied Sciences, 2018.

[39] S. private and public organisations. Rgs: Referentie grootboekschema. https://www.referentiegrootboekschema.nl/, 2022.

[40] F. Rayhan, S. Ahmed, A. Mahbub, R. Jani, S. Shatabda, and D. M. Farid. Cusboost: Cluster-based under-sampling with boosting for imbalanced classification. In 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), pages 1–5. IEEE, 2017.

[41] J. N. v. Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren. Fast algorithm selection using learning curves. In International symposium on intelligent data analysis, pages 298–309. Springer, 2015.

[42] C. Romero and S. Ventura. Data mining in education. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 3(1):12–27, 2013.

[43] N. F. Rusland, N. Wahid, S. Kasim, and H. Hafit. Analysis of naïve bayes algorithm for email spam filtering across multiple datasets. In IOP conference series: materials science and engineering, volume 226, page 012091. IOP Publishing, 2017.

[44] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

[45] Y. Shao, S. Taylor, N. Marshall, C. Morioka, and Q. Zeng-Treitler. Clinical text classification with word embedding features vs. bag-of-words features. In 2018 IEEE International Conference on Big Data (Big Data), pages 2874–2878. IEEE, 2018.

[46] A. software. Erp-systems for your company. https://www.afas.nl/, 2022.

[47] B. Strang, P. v. d. Putten, J. N. v. Rijn, and F. Hutter. Don't rule out simple models prematurely: a large scale benchmark comparing linear and non-linear classifiers in openml. In International Symposium on Intelligent Data Analysis, pages 303–315. Springer, 2018.

[48] A. Sun, E.-P. Lim, and Y. Liu. On strategies for imbalanced text classification using svm: A comparative study. Decision Support Systems, 48(1):191–201, 2009.

[49] P.-N. Tan, M. Steinbach, and V. Kumar. Introduction to data mining. Pearson Education India, 2016.

[50] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(1):281–288, 2008.

[51] G. Wahba et al. Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. Advances in Kernel Methods-Support Vector Learning, 6:69–87, 1999.

[52] L. Wang. Support vector machines: theory and applications, volume 177. Springer Science & Business Media, 2005.