Technische Hochschule Deggendorf
Fakultät Angewandte Informatik
Studiengang Master Artificial Intelligence and Data Science

Textklassifizierung von Kundenanfragen via E-Mail

Text classification of customer inquiries via e-mail

Masterarbeit zur Erlangung des akademischen Grades:
Master of Science (M.Sc.)
an der Technischen Hochschule Deggendorf

Vorgelegt von:	Prüfungsleitung:
Armin Weigold	Prof. Dr. Fischer
Matrikelnummer: 00629413	

Am: 05. Mai 2023

TECHNISCHE HOCHSCHULE DEGGENDORF

Erklärung

Name des Studierenden: Armin Weigold

Name des Betreuenden: Prof. Dr. Fischer

Thema der Abschlussarbeit:

Textklassifizierung von Kundenanfragen via E-Mail . . . .

1. Ich erkläre hiermit, dass ich die Abschlussarbeit gemäß § 35 Abs. 7 RaPO (Rahmenprüfungsordnung für die Fachhochschulen in Bayern, BayRS 2210-4-1-4-1-WFK) selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

   Deggendorf, 01.05.2023 Aei

   Datum Unterschrift des Studierenden

2. Ich bin damit einverstanden, dass die von mir angefertigte Abschlussarbeit über die Bibliothek der Hochschule einer breiteren Öffentlichkeit zugänglich gemacht wird:

   ☒ Nein

   ○ Ja, nach Abschluss des Prüfungsverfahrens

   ○ Ja, nach Ablauf einer Sperrfrist von . . . Jahren.

   Deggendorf, 01.05.2023 Aei

   Datum Unterschrift des Studierenden

Bei Einverständnis des Verfassenden vom Betreuenden auszufüllen:

Eine Aufnahme eines Exemplars der Abschlussarbeit in den Bestand der Bibliothek und die Ausleihe des Exemplars wird:

○ Befürwortet

○ Nicht befürwortet

Deggendorf, ...

Datum Unterschrift des Betreuenden

Abstract

This work addresses a problem of text classification. The research question focuses on identifying the optimal workflow to solve the given problem. The dataset consists of roughly 430,000 labeled e-mails. The problem is tackled in two steps, namely vectorizing the text and applying a classification algorithm. Several algorithms, including Word2vec and Tf-idf for vectorization, and Random Forest, Support Vector Machine, Graph Neural Network, and Feed-Forward Neural Network for classification, were evaluated. The results show that Word2Vec performed well, while Tf-idf had too high memory demands. In terms of classification, the Feedforward Neural Network achieved the highest F1 scores of 0.89-0.90 depending on the trial, followed by Random Forest and Support Vector Machine with F1 scores of 0.87-0.89, while the graph neural network achieved F1 scores of 0.80-0.87.

Contents

Table of figures

List of tables

1 Introduction

The company CHECK24 Vergleichsportal Energie GmbH offers customers a platform to compare electricity or gas tariffs from different providers. It also offers the service of simplifying the conclusion, changes and terminations of contracts.

Communication with customers and providers takes place primarily via e-mail. However, processing all these inquiries involves a great deal of work. In order to reduce this, the e-mails should be preprocessed automatically. For example, messages can be sorted by subject, prioritized, a suitable response template can be specified or they could be processed automatically.

From sorting in different categories of topics, a multiclass classification problem can be derived. However, some difficulties exist. First of all, customers' e-mails are very diverse and don't follow a given structure. Moreover, most e-mails can not be meaningfully assigned to a class. For this reason, there is an "Other" class that includes all data points that cannot be reasonably assigned. Almost 75% of the dataset is assigned to that class. Also, the remaining dataset is unbalanced, so some classes are over- or under-represented. In addition, the boundaries between some classes are fluid, so even human labeling can have errors. These circumstances add further complexity to the problem.

1.1 Goal of this work

The goal of the work is to develop a workflow that solves the classification problem above. The problem can be broken down into two main steps. First, the plain text must be encoded into a machine-readable form. There are several possible approaches for this. After that, the transformed messages have to be classified using a machine learning model. In this step, too, there are various models to choose from. In addition, there are parameters within a model that can be optimized. Ultimately, the combination of approaches and parameters that has the highest practicability, performance and quality of the classification should be determined. Hence the central question, this work aims to answer is: Which workflow solves the upper classification problem best?

1.2 Structure of this work

The structure of this work is as follows: Firstly related work is mentioned. Thereafter the theoretical background is illuminated. For both vectorization and classification, all applied algorithms are named and their functionality explained. The next chapter explicates the experimental setup. Firstly the dataset and its procession is described. Secondly the implementation of the vectorization is depicted. Thirdly the application of the classification algorithms are expounded. Thereafter the results are evaluated. Firstly the results of every model are depicted. Then the models are compared to each other. In the last chapter those results are discussed.

2 Related work

Text classification is one of the core problems of natural language processing (Huang et al., 2019). It has already been used successfully in many problems (Kowsari et al., 2019). Examples include medical texts (Qing et al., 2019) and accident prevention in industry (Sánchez-Pi et al., 2014). E-mail data have also been dealt with in the literature. However, most research is aimed at spam or phishing detection (Verma et al., 2020).

As possibilities for vectorization or feature extraction Kowsari et al. name the following methods: Bag of Words (BoW), Term Frequency-Inverse Document Frequency (Tf-idf), and Word Embedding methods such as Word2Vec., Continuous Bag-of-Words or Continuous Skip-Gram algorithms (Kowsari et al., 2019).

There are also various approaches for the actual text classification. Well-known machine learning models have been successfully applied. Pranckevičius and Marcinkevičius and Ikonomakis et al. mention Naive Bayes, Random Forest, Decision Tree, Support Vector Machines as well as Logistic Regression Classifiers (Ikonomakis et al., 2005; Pranckevičius and Marcinkevičius, 2017). With the support vector machine and the random forest Kim et al. and Bouaziz et al. also brought traditional classification algorithms to use (Bouaziz et al., 2014; Kim et al., 2005).
In recent years more sophisticated deep learning models became popular for text classification (Minaee et al., 2021). These include graph neural networks from Huang et al., convolutional neural networks from Jacovi et al. and Liang et al.
and attention models from Sun and Lu (Huang et al., 2019; Jacovi et al., 2020; Liang et al., 2019; Sun and Lu, 2020). This work will deal with machine learning as well as deep learning approaches.

In contrast to most relevant research, the dataset used in this work is in German. Because of unique features of the German language like compound words, the data might behave differently than expected.

3 Theoretical background

This chapter explains the theoretical background of the algorithms and techniques used in this work. The first section deals with the vectorization of the data, the second one with the actual classification.

3.1 Vectorization

In order to train and utilize a machine or deep learning classifier on the e-mail data, the natural language must be transformed into a machine readable vector. Therefore two different approaches are applied. Firstly Term Frequency-Inverse Document Frequency (Tf-idf), a simple algorithm, and secondly Word2vec, a more sophisticated approach.

3.1.1 Tf-idf

The Tf-idf algorithm assigns weights to each word in a corpus based on its frequency and importance in the text (Sparck Jones, 1972).

The algorithm works by first calculating the term frequency (tf) of each word in a corpus, which is simply the number of times the word appears in the text. This is then followed by calculating the inverse document frequency (idf) of each word, which is based on the number of documents that contain the word. Specifically, the tf-idf value of a word in a document is calculated:

$$W(d,t) = TF(d,t) * log(N/df(t)),$$

where N is the number of documents and df(t) is the number of documents containing the term t in the corpus (Kowsari et al., 2019).

This formula gives a higher weight to words that are both frequent in the document and rare in the corpus. This weight can be used to represent the importance of each word in the context of the corpus.

3.1.2 Word2Vec

The word2vec algorithm is a neural network-based approach for generating word embeddings, which are numerical representations of words that capture their meaning and context. It involves training a neural network on a large corpus of text by feeding pairs of words or "word contexts" into the network and adjusting the weights of the connections between neurons using negative sampling (Kowsari et al., 2019). To this end the models Continuous Bag-of-Words and Continuous Skip-Gram are utilized.

Continuous Bag-of-Words model predicts the current word given its surrounding context. For example, given the sentence "I like to cancel my electricity contract", the Continuous Bag-of-Words model would predict the word "cancel" based on the surrounding words "I", "like", "to", "my", "electricity" and "contract".

The Skip-gram model works the opposite way by predicting the surrounding context given a current word. For instance, given the word "cancel", the Skip-gram model would predict the surrounding context of "I", "like", "to", "my", "electricity" and "contract" (Mikolov et al., 2013).
The resulting high-dimensional vectors represent the meaning and context of words in the vocabulary.

3.2 Classification

After vectorization the data can be processed by classification algorithms. The classifiers applied in this work are described in this section.

3.2.1 Random Forest

A random forest classifier is an ensemble learning method used for classification tasks. It is a collection of decision trees, where each decision tree is trained on a different subset of the training data and using a different set of features, selected randomly.

The following are the basic steps in the random forest algorithm:
Firstly a subset of the training data is selected at random from the original dataset.
On this subset of the data a decision tree is created. Therefore the data are recursively split based on the most discriminative features, until the tree reaches a stopping criterion.

The most discriminative feature can be determined by the Gini Index. It is calculated by the formula:

$$Gini = \sum_{k=1}^{K} P_k(1 - P_k),$$

where K is the number of classes and P_k is the proportion of samples that belong to class k. Considering a highly discriminative feature, P_k approximates 1 or 0, which leads to the

Gini Index approximating 0. Hence the feature with the lowest Gini Index is chosen for the split (James et al., 2013).

This procedure is repeated multiple times to create several decision trees.
To classify a new instance, as shown in figure 1, each of the decision trees are used to predict the class of the instance. The class with the most votes across all the trees is the final predicted class for the instance.

By randomly selecting subsets of the training data and features, and creating multiple decision trees, the random forest algorithm helps to reduce overfitting and increase generalization. It also enables the algorithm to handle missing data and noisy features (Breiman, 2001; Kowsari et al., 2019).

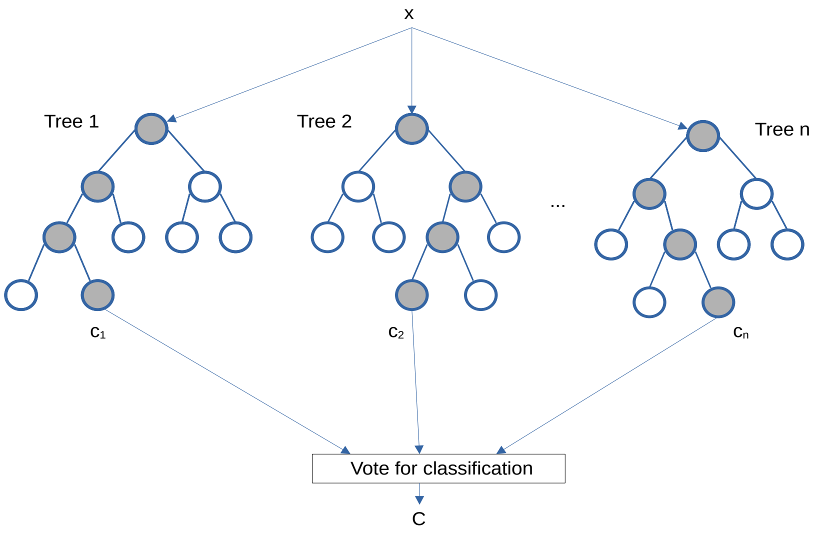


Figure 1: Classification via Random Forest

3.2.2 Support Vector Machine

Support Vector Machines are a class of machine learning algorithms used for both classification and regression analysis. Support Vector Machines were originally developed for binary classification problems, but have since been extended to handle multi-class classification problems (Boser et al., 1992).

Support Vector Machines work by finding the hyperplane that best separates the different classes of data points. The hyperplane is chosen such that it maximizes the distance between the closest data points from each class. These closest data points are called support vectors. Formally, let $[(x_1, y_1), ..., (x_n, y_n)]$ be a training set of n instances, where $xi \in Rd$ is a d-dimensional feature vector and $y_i \in (-1, 1)$ is the corresponding class label. The goal of a Support Vector Machine is to find a hyperplane that separates the two classes, which can be represented by the equation:

$$wx + b = 0,$$

where w is a weight vector that determines the orientation of the hyperplane, b is a bias term that shifts the hyperplane away from the origin, and · denotes the dot product. The hyperplane separates the data points into two regions: one region for each class as shown in figure 2.

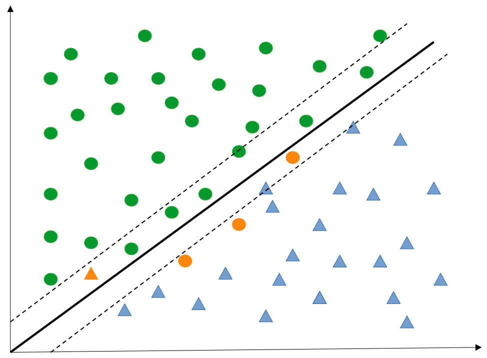


Figure 2: Linear classification via Support vector machine

To determine the optimal hyperplane, Support Vector Machines solve an optimization problem that minimizes a loss function subject to constraints. A common loss function for Support Vector Machines is the hinge loss function. It is defined as follows:

$$l = max(0, 1–t * y)$$

Where t is the expected output which equals ±1 and y is the classification output.

Consequently, the cost function penalizes misclassification errors relatively to the distance to the decision threshold. Furthermore it encourages the hyperplane to have a large margin on correct classifications. The constraints ensure that the hyperplane separates the data points correctly (Rosasco et al., 2003; Vapnik and Chervonenkis, 1964).

There are several approaches to achieve multi-class classification. The most common ones are One-versus-one, One-versus-all or One-versus-the-rest and Crammer-Singer.

In the One-versus-one approach, the Support Vector Machine algorithm trains a separate binary classifier for each pair of classes. Each classifier separates the data of one class from the data of the other class in the pair.

In the One-versus-all approach however, a distinct binary classifier is trained for every class. Each classifier separates the data of one class from the data of all other classes. For both approaches the new data point is classified by each binary classifier, and the class with the most votes is chosen as the predicted class during the testing phase (Kowsari et al., 2019).

Unlike One-versus-one and One-versus-all the Crammer-Singer algorithm trains a single multi-class classifier that directly optimizes a joint objective function.

Therefore the Support Vector Machine algorithm learns a linear mapping from the feature space to a space of dimensionality equal to the number of classes. The decision function is then defined as the argmax of the linear scores (Crammer and Singer, 2001).

3.2.3 Neural Networks

Neural networks are a type of machine learning algorithm inspired by the structure and function of the human brain (Hopfield, 1982). They consist of layers of interconnected neurons that receive input data and perform computations on it.

The simplest type of neural network is the Feed Forward Neural Network. This type of network consists of an input layer, one or more hidden layers, and an output layer, as shown in figure 3. Each layer contains a set of neurons, which receive input from the previous layer and produce output for the next layer. The connections between neurons are represented by weights, which are learned during training.

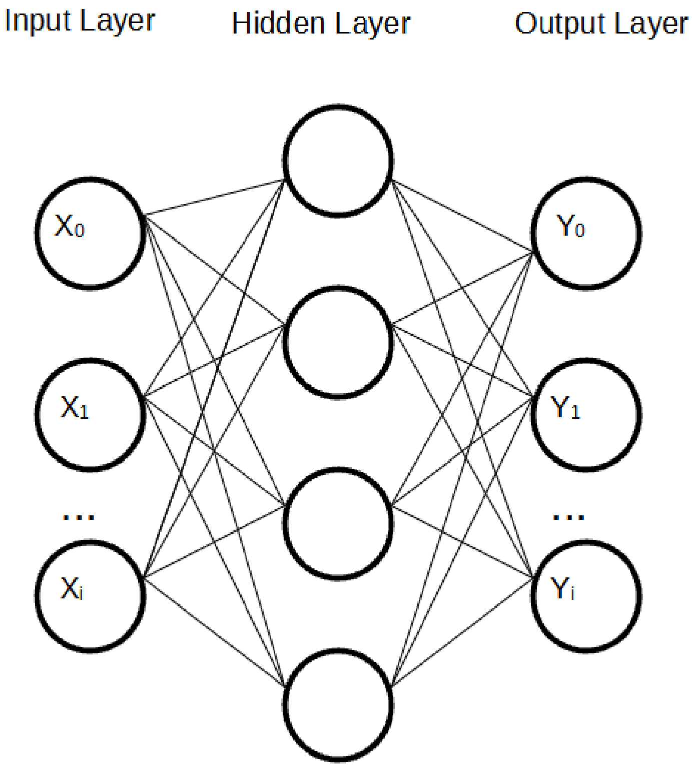


Figure 3: Fully connected Feed Forward Neural Network

Training a neural network involves adjusting the weights to minimize a cost function, which measures the difference between the network's predictions and the true class labels. This is typically done using an optimization algorithm such as stochastic gradient descent. During training, the network learns to recognize patterns in the input data that are predictive of the class labels.

One important aspect of neural network training is the activation function. The activation function is applied to the output of each neuron, and determines the throughput to the next layer. Activation functions used in this work include the softmax function and the rectified linear unit (ReLU) function.

Relu takes the form of

$$f(x) = max(0, x)$$

Therefore it returns the input value if it is positive and 0 otherwise. The ReLU function is computationally efficient and has been shown to be effective in deep neural networks (Nair and Hinton, 2010).

The Softmax activation function is commonly used in the output layer of a neural network for multi-class classification tasks. It takes a vector of real-valued inputs and transforms them into a vector of probabilities that sum to 1. The Softmax function is given by:

$$f(x) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$$

where i = 1,...,K, K is the number of classes and x_i is the input value for class I (Bridle, 1990).

Neural networks can also be regularized to prevent overfitting, which occurs when the network memorizes the training data rather than learning generalizable patterns. A common regularization technique used in this work is called dropout.

It works by randomly deactivating or so to speak "dropping out" some fraction of the input or hidden units during training. The idea behind dropout is to force the network to learn redundant representations of the input, which makes it more robust to noise and improves generalization performance.

During training, dropout works by randomly setting some fraction of the input or hidden units to 0 with a probability p. The remaining units are then scaled by a factor of

$$\frac{1}{1-p}$$

to maintain their expected value. During testing, all units are used and no scaling is applied (Goodfellow et al., 2016; Srivastava et al., 2014).

3.2.4 Graph Neural Networks

Graph Neural Networks are a type of neural network that operate on graph-structured data. Graphs are represented by nodes which are connected by edges. Therefore this data structure that can represent complex relationships between entities. Graph Neural Networks are designed to leverage this structure to learn useful representations of the nodes and edges in a graph (Scarselli et al., 2009).

Natural language can also be represented as a graph. In that case words are represented as nodes with edges connecting adjacent ones. Embeddings created with the Word2Vec algorithm only carry the meanings of the words themselves, while Text graphs also carry the information of the context between words.

The sentences “The meeting was bad but the dinner was good” and “The meeting was good but the dinner was bad” serve as an example. The vector representation of these two sentences generated by the Tf-idf or Word2vec algorithm are indistinguishable. A Graph representation however preserves the information of which adjective refers to which noun.

The first step in processing graphs with neural networks is to represent the graph data in a way that can be easily processed by the network. One approach is to represent the graph as an adjacency matrix, where each entry in the matrix indicates whether there is an edge between two nodes in the graph. Another approach is to represent the graph as a set of node and edge features, where each node feature vector corresponds to the attributes of the node, and each edge feature vector corresponds to the attributes of the edge. Since the edge attributes contain the nodes, which the edges are connecting, a graph can be represented (Huang et al., 2019).

Once the graph is represented, it can be processed by a neural network using graph convolution. Graph convolution is similar to regular convolution, but operates on the graph structure instead of the regular grid structure. The basic idea behind graph convolution is to define a set of learnable filters that are applied to the node features and their neighboring nodes to generate new node features. This process can be repeated multiple times to incorporate information from more distant nodes in the graph (Wu et al., 2021).

4 Experimental setup

This chapter describes the experimental setup. First of all the Dataset is described in detail. Then the steps of vectorization as well as classification are explained.

4.1 Description and preprocessing of the Dataset

The dataset consists of 433,220 E-Mails sent by customers, which were labelled by customer support during the processing of each e-main. The labels represent 10 classes of e-mails: *OTHER, REVOCATION, BONUS, CHANGE_INFO, STATUS_QUESTION, ACCOUNT, CANCELLATION, INSPECT_DOCUMENTS, FEEDBACK,* and *TEST.* The distribution of the Labels is shown in table 1.

Table 1: Distribution of classes

OTHER	307,933
REVOCATION	34,361
BONUS	32,675
CHANGE_INFO	30,017
STATUS_QUESTION	24,626
ACCOUNT	1,693
CANCELLATION	795
INSPECT_DOCUMENTS	610
FEEDBACK	505
TEST	5

Only a fraction of e-mails can be labeled unambiguously. All remaining instances are therefore labelled *OTHER.* Furthermore the list of classes is not exhaustive, therefore messages that treat a topic without an associated class, are also labeled *OTHER*.
Some classes have too little support to be considered in the evaluation. That way the relevant classes are R*EVOCATION, BONUS, CHANGE_INFO and STATUS_QUESTION.*

E-mails in which a client wants to revoke their contract are labeled as *REVOCATION.* Suppliers often offer a bonus for new customers. E-Mails related to this topic fall under the class *BONUS.* In Messages labeled as *CHANGE_INFO* clients provide Information vital to the change of suppliers, for instance their electricity meter reading. The Class *STATUS_QUESTION* contains E-Mails in which customers ask a question about their current contract.

This is an example for an e-main labelled as REVOCATION:

“Liebes Check 24 Team,
hiermit möchte ich den Auftrag für den Stromvertrag stornieren.
Auftragsnummer: 69481966”

English translation:
“Dear Check 24 team,
I would like to cancel the order for the electricity contract.
Order number: 69481966”

This text data has been processed in the following way. The words have been reduced to their stems, which means that the endings of the words have been removed to obtain the root form of each word. This process is called “stemming” and was executed via (“NLTK :: nltk.stem.snowball module,” n.d.). For example, the stem of "stornieren" would be "storni". Moreover stop words have been removed from the text. Stop words are common words that do not convey significant meaning like “ich” and ”für”. Punctuation marks such as periods, commas, and exclamation points have also been removed. Furthermore all words have been converted to lower case. This is done to ensure that words that are the same except for their capitalization are treated as the same word (Kowsari et al., 2019). After this preprocessing the example above looks like this:

“lieb check 24 team hiermit mocht auftrag stromvertrag storni auftragsnumm 69481966”

4.2 Vectorization

In the next step the dataset needs to be vectorized. The corpus consists of 433,220 messages, which consist of a total of 13,645,398 and 529,787 distinct tokens.

4.2.1 Tf-idf

For Tf-idf vectorization the Sklearn function is used (“sklearn.feature_extraction .text.TfidfVectorizer,” n.d.).However with the application of the Tf-idf algorithm however arouse an issue. One vector representing one e-mail has the length of the number of distinct tokens. A vector of this length requires approximately 2.5 megabytes of memory. Scaled up to the whole corpus, all available data would take up approximately 1 terabyte of memory. Due to this impracticability, Tf-idf is no longer considered in this work.

4.2.2 Word2vec

Word2vec was applied by using the model form the library Gensim (“models.word2vec – Word2vec embeddings — gensim,” n.d.). The entire corpus was used to train the model. After training, each token was mapped to a 100-component vector representation. The resulting token vectors were then averaged for each e-mail to obtain a fixed-length vector representation of each e-main. This representations were used as input features for the

Random Forest, the Support Vector Machine and the Feed Forward Neuronal Network Models.

4.3 Classification

This section describes the application of all classification algorithms used in this work. Every classifier setup is trained and tested twice. Firstly with all data in order to test the recognition of relevant classes against unclassified Mails falling into the *OTHER* class. Secondly with all data excluding the *OTHER* class to evaluate the ability to distinguish the relevant classes among themselves. Instances of irrelevant classes are included in the training and test sets but due to relatively low support they only have negligible impact on the results. They are therefore ignored in the upcoming evaluation.

4.3.1 Random Forest and Support Vector Machine

For both the Random Forest (“sklearn.ensemble.RandomForestClassifier — scikit-learn 1.2.0 documentation,” n.d.) and Support Vector Machine (“sklearn.svm.LinearSVC,” n.d.) trial the library Scikit-learn’s (Pedregosa et al., n.d.; “sklearn.ensemble.RandomForestClassifier — scikit-learn 1.2.0 documentation,” n.d.) implementation was utilized.

Hyperparameters were optimized using a half grid search (Bahi, 2021; “sklearn.model_selection.HalvingGridSearchCV,” n.d.). This resulted in the following values: Max_depth = None, min_samples_split = 5, n_estimators = 200 for Random Forest and

max_iter = 5000, tol = 0.01, multi_class = 'crammer_singer' for Support Vector Machine.

4.3.2 Graph Neural Network

Graph Neural Networks require data with a graph structure as input. Such graphs can be obtained from text data. Therefore each token is represented by an embedding generated by the Word2vec algorithm. These embeddings however are not averaged on the message level, but serve as nodes in the text graph. Sequential words are connected by edges as depicted in figure 4.

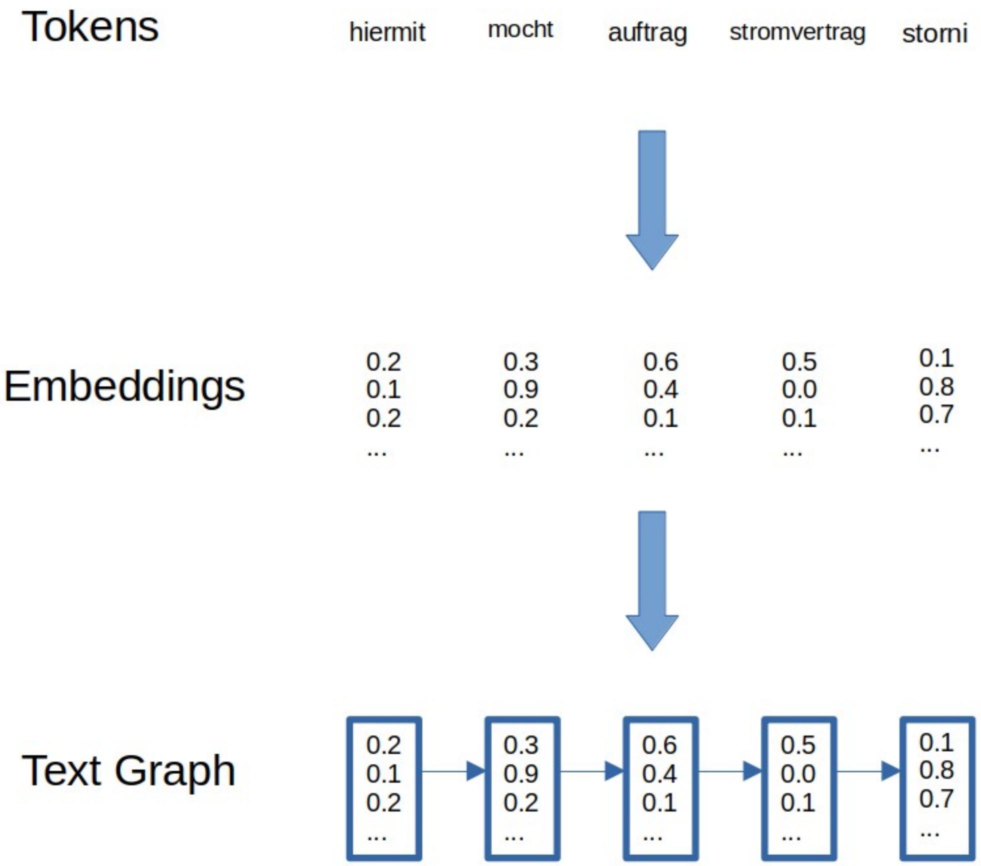


Figure 4: Workflow of generating a text graph

To implement the Graph Neural Network, the library StellarGraph was utilized (CSIRO's Data61, 2018). The Graph Neural Network consists of two Graph Convolution Layers, with the tanh activation function and the first one having a size of 32 the last one of 1. For one specific trial two more Graph Convolution layers with a size of 32 are added.

After the message passing process the Text Graphs are fed into a 1D Convolutional Neural Network. It consists of a 1D convolutional layer with 16 filters and a kernel size and strides equal to the sum of the layer sizes of the Graph Convolution Layers, a 1D max pooling layer with a pool size of 2, a second 1D convolutional layer with 32 filters and a kernel size of 5, with strides of 1, a flatten layer, a fully connected layer with 128 units and a ReLU activation function, a dropout layer with a rate of 0.5 and a final dense layer with number of classes units and a softmax activation function. This architecture was inspired by Zhang et al., 2018.

4.3.2.1 Limitations

Due to the larger size of text graphs compared to averaged embedding and their more complex architecture, the Graph Neural Network algorithm requires significantly more computation resources.

Table 2: duration of computations

Computation	Number of Data Instances	Duration (min)
Building Text Graphs	20000	3:05
Training the Graph Network and Predicting	20000	10:55
Building Text Graphs	50000	7:19
Training the Graph Network and Predicting	50000	33:37

Training the Random Forest and Predicting	433220	11:53
Training the Support Vector Machine and Predicting	433220	14:35
Training the Feed Forward Network and Predicting	433220	3:10

Table 2 shows that the Graph Neural Network processes less data in the same amount of time than other algorithms considered. Therefore a smaller Dataset is used for Graph Neural Networks in the upcoming evaluation.

4.3.2.2 Adaptations

Graph Neural Networks are more complex, i.e. have more adaptable parameters than the other classifiers applied in this work. Therefore several trials will be conducted with Graph Neuronal Networks to look into the effect of adapting this parameters.

Firstly the Data available to the Graph Neural Network algorithm is increased. The Graph Network has access to less data than the other classifiers, owed to the high computational cost. Hence the impact of the amount of data is a useful information to gain.

An approach to reduce the computation effort is to reduce the length of the enbeddings in the text graph. The impact of this measure on the classification is tested in the next trial.

In order to optimize the input, the text graphs are augmented for the next trial. Therefore edges connect not just the next but the next but one token as shown in figure 5.

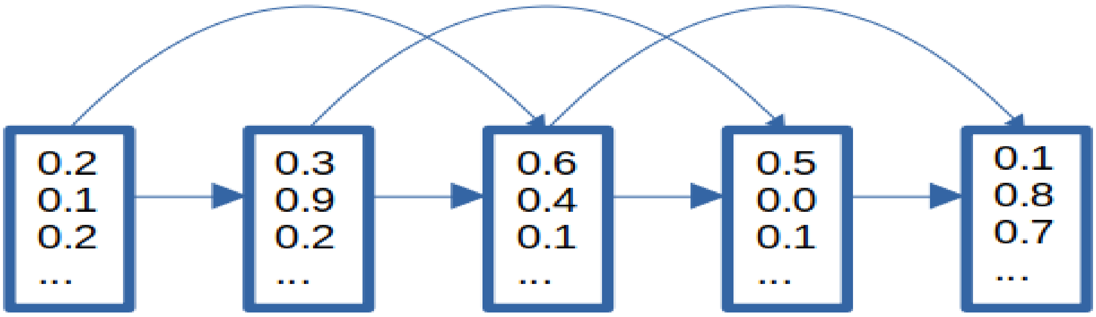


Figure 5: Text graph with edge range = 2

To further examine the impact of the graph convolution, the amount of graph convolution layers is doubled for the last trial involving Graph Neuronal Networks.

4.3.3 Feed Forward Neural Networks

The Feed Forward Network is designed to mimic the Graph Network without the graph convolution. Therefore the Graph Convolution as well as the 1D Convolution layers are replaced with another fully connected and a dropout layer.

So the network consists of a fully connected layer with 128 units and a ReLU activation function a dropout layer with a rate of 0.5, another pair of dense and dropout layers with the same parameters and a dense layer with number of classes units and a softmax activation function.

5 Evaluation

For the evaluation a test set of 20% was utilized. Therefore every data instance had a 20% chance to go in the test set and an 80% chance to be used in training. That way test data tend to be spread equally among the dataset. For the evaluation of the classification the following metrics are considered: The precision measures the proportion of true positive predictions out of all positive predictions, the recall is calculated by dividing the number of true positive predictions by the amount of all actual positive observations, the F1 score combines precision and recall. It is calculated according to this formula (Lever et al., 2016):

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

Those metrics only evaluate the classification of a single class. In order to depict the Multi-class problem as a whole, further metrics are considered: The accuracy measures the proportion of correct predictions out of all predictions and the weighted average F1 score, which is calculated as follows:

$$Weighted average F1 score = \frac{F1C_1 * supportC_1 + F1C_2 * supportC_2 + ...}{*supportC_1 + *supportC_2 ...}$$

The calculation of the metrics is automated by the function (“sklearn.metrics.classification_report,” n.d.)

5.1 Random Forest

The Random Forest Classifier (RDF) achieves an accuracy of 0.85 and a weighted F1 average of 0.87 on a test set of 86526 instances. The *OTHER* class is best recognized with an F1 score of 0.90. The F1 Score of other relevant classes is between 0.53 and 0.77.

The confusion Matrix (figure 6) shows most false classification resolving around the *OTHER* class. A lot of *OTHER* messages are falsely classified in relevant classes which takes the relevant classes precision down. Messages labeled as *OTHER* end up misclassified relatively rarely. Hence *OTHERs* precision and relevant classes recall are comparably high. The classes *CHANGE_INFO* and *STATUS_QUESTION* are most affected. Hence their precision falls under 0.5. This accumulation of misclassifications could be explained by fuzzy separation between classes. An e-mail labeled *OTHER* could easily be misclassified as a relevant class if its contents are similar to e-mail labeled as said class.

Table 2: Results RDF Classifier

Class	Precision	Recall	F1-score	Support
BONUS	0.63	0.89	0.74	4664
CHANGE_INFO	0.46	0.87	0.60	3194
OTHER	0.98	0.84	0.90	71413
REVOCATION	0.67	0.91	0.77	5067
STATUS_QUESTION	0.38	0.86	0.53	2171
Weighted average	0.91	0,85	0.87	86526
Accuracy			0.85	86526

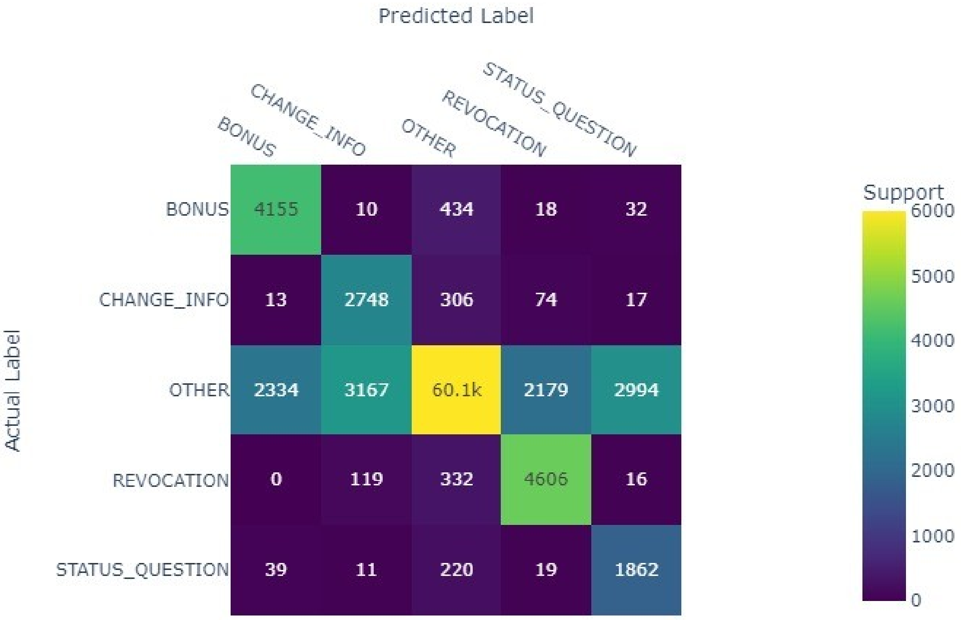


Figure 6: Confusion matrix RDF

If the *OTHER* class is excluded from the dataset, the accuracy amounts to 0.88 and the weighted F1 average to 0.89. The F1 scores of the relevant classes increase to the range of 0.87 and 0.92. According to the confusion matrix (figure 7), the classes *REVOCATION* and *CHANGE_INFO* are most frequently confused with each other. A possible reason for this observation is, the semantic proximity of the two classes. Both *REVOCATION* and *CHANGE_INFO* involve the termination of a contract. STATUS_QUESTION has the worst F1 score but the misclassified data is equally divided between the classes.

Table 3: Results RDF Classifier without OTHER class

Class	Precision	Recall	F1-score	Support
BONUS	0.91	0.93	0.92	6360

CHANGE_INFO	0.89	0.86	0.88	6222
REVOCATION	0.92	0.86	0.89	7258
STATUS_QUESTION	0.89	0.86	0.87	5197
Weighted average	0.90	0.88	0.89	25081
Accuracy			0.88	25081

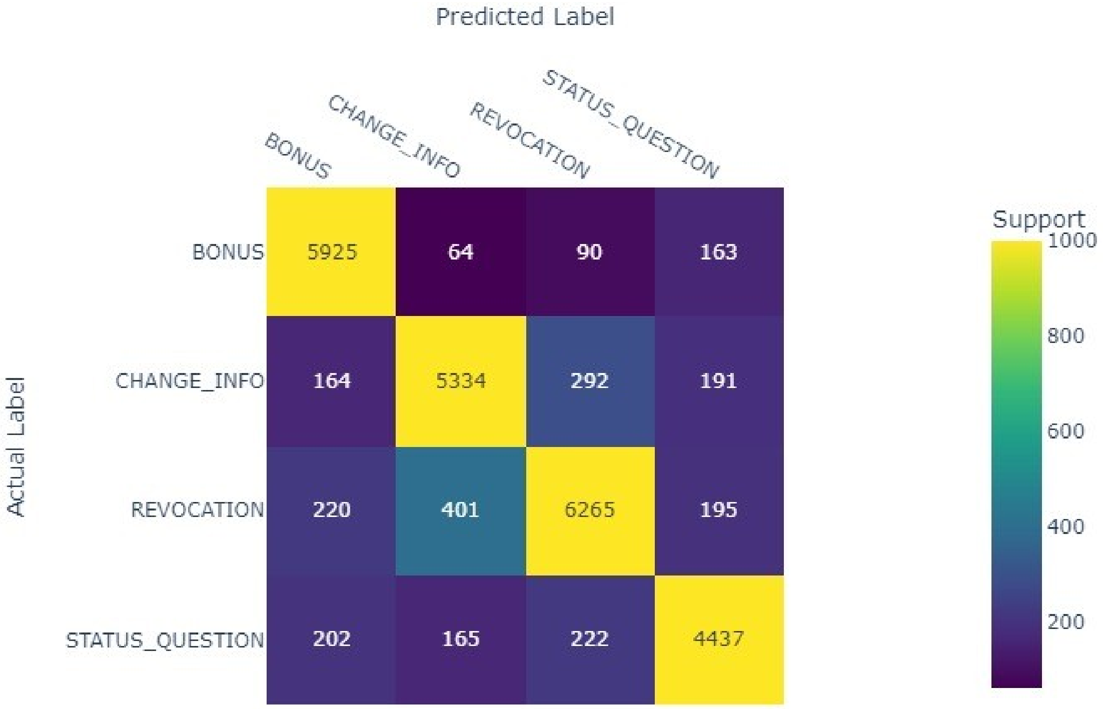


Figure 7: Confusion Matrix RDF excluding OTHER

5.2 Support Vector Machine

The Support Vector Machine (SVM) scores an accuracy of 0.85 and an F1 weighted average of 0.87 on the same test. The *OTHER* class was recognized best with the highest F1 score of 0.91, while the F1 measure for other relevant classes ranged from 0.58 to 0.80.

The confusion matrix (figure 8) yields similar information as in the previous trial. Again many instances labeled *OTHER* are classified as relevant classes.

Table 4: Results SVM Classifier

Class	Precision	Recall	F1-score	Support
BONUS	0.73	0.85	0.79	5598
CHANGE_INFO	0.45	0.84	0.58	3170
OTHER	0.96	0.86	0.91	68911
REVOCATION	0.74	0.87	0.80	5983
STATUS_QUESTION	0.46	0.78	0.58	2857
Weighted average	0.90	0.85	0.87	86524
Accuracy			0.85	86524

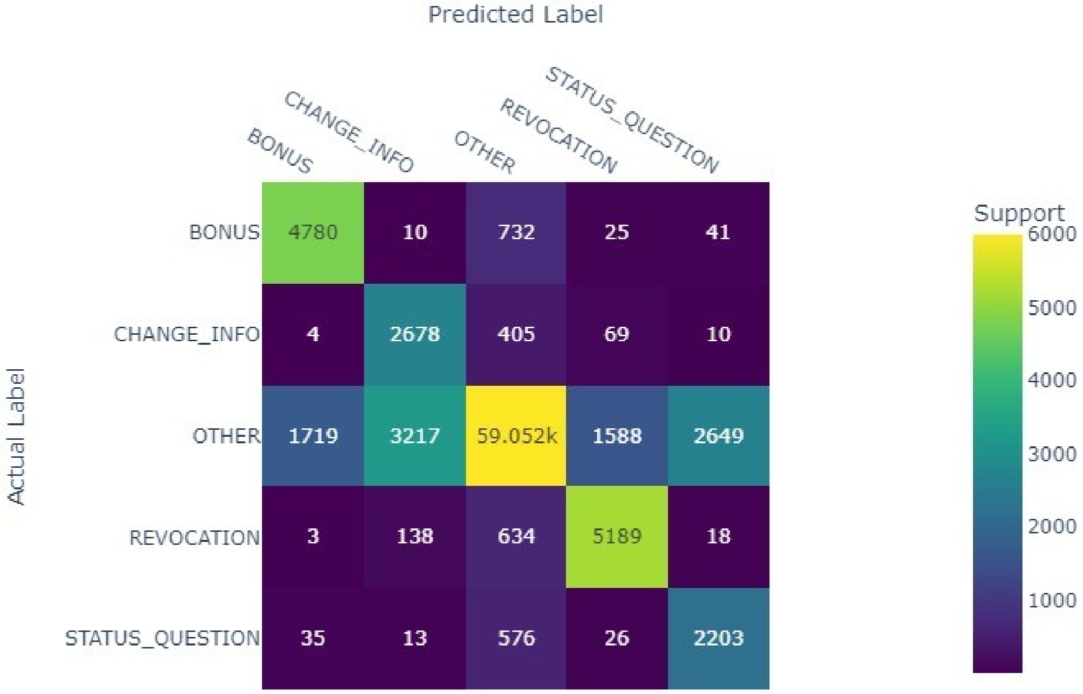


Figure 8: Confusion matrix SVM

In a trial under exclusion of the *OTHER* class, the accuracy and F1 weighted average is 0.89. The F1 scores of the relevant classes are then between 0.86 and 0.94, which is a slightly large spread between classes compared to the Random Forests results. In the confusion matrix (figure 9) similar observations as in the corresponding Random Forest trial can be made.

Table 5: Results SVM Classifier without OTHER class

Class	Precision	Recall	F1-score	Support
BONUS	0.93	0.94	0.94	6896
CHANGE_INFO	0.87	0.89	0.88	5901
REVOCATION	0.91	0.91	0.91	6810

STATUS_QUESTION	0.91	0.81	0.86	5113
Weighted average	0.90	0.89	0.89	25081
Accuracy			0.89	25081

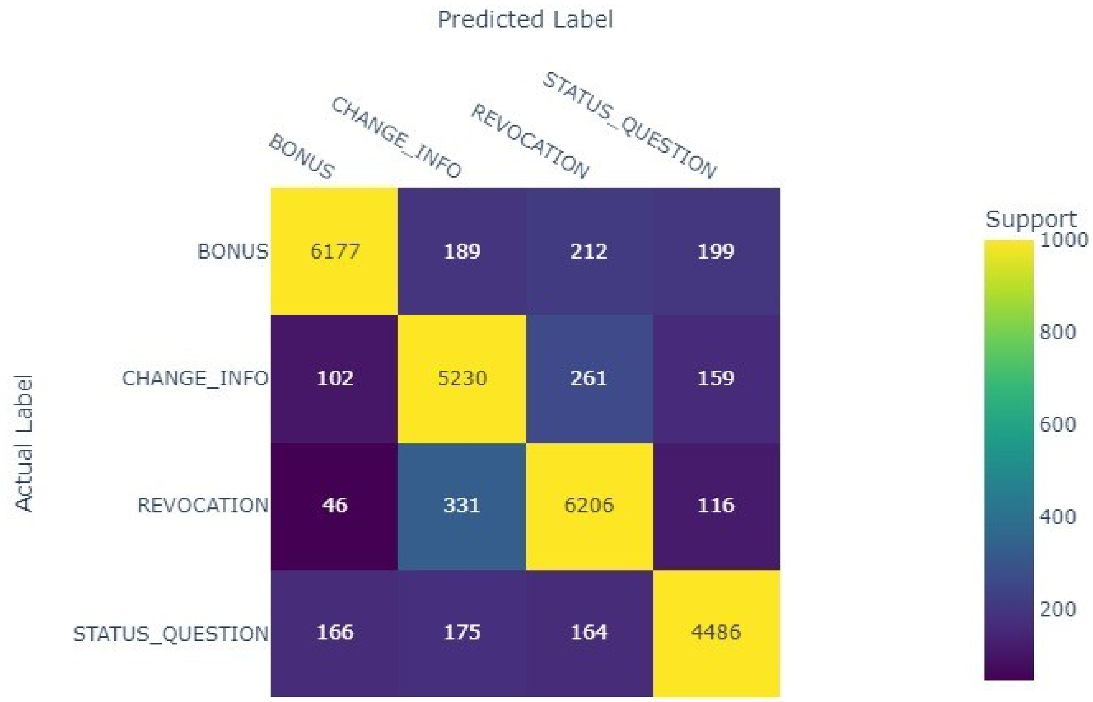


Figure 9: Confusion matrix SVM excluding OTHER

5.3 Graph Neural Networks

The Graph Neural Network (GNN) achieves an accuracy of 0.78 and a weighted F1 average of 0.79 on a data set of 16000 training and 4000 test instances. The OTHER class is best recognized with an F1 score of 0.85. The F1 Score of other relevant classes is between 0.56 and 0.76. As with other algorithms, misclassifications involve the *OTHER* class for the most part. In contrast to previous trials, this confusion goes both ways. As depicted in Figure 10, instances of OTHER are misclassified as relevant classes as well as the other way around.

Table 6: Results GNN 20000 data instances

Class	Precision	Recall	F1-score	Support
BONUS	0.63	0.81	0.71	313
CHANGE_INFO	0.51	0.63	0.56	240
OTHER	0.90	0.80	0.85	2922
REVOCATION	0.74	0.78	0.76	296
STATUS_QUESTION	0.52	0.73	0.61	273
Weighted average	0.80	0.78	0.79	4121
Accuracy			0.78	4121

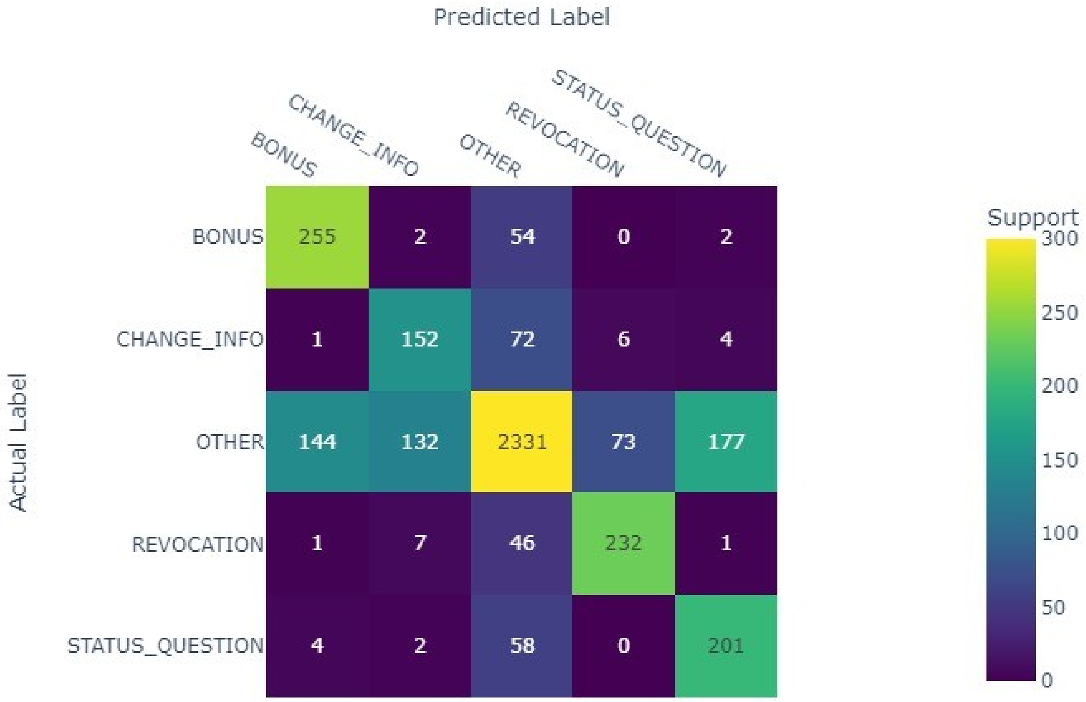


Figure 10: Confusion Matrix GNN 20000 data instances

The next trial consisted of about 40000 trainings and 10000 test instances. The results showed an accuracy of 0.77 and a weighted average F1 score of 0.79. The F1 score of the relevant classes ranged from 0.63 to 0.79 when the *OTHER* class scored highest again. The confusion matrix (figure 11) shows a familiar picture. Similar as in previous Trials instances labeled *OTHER* are again often classified as one of the relevant classes. Misclassifying members of relevant classes as *OTHER* is not as common as in previous trials.

Table 7: Results GNN 50000 data instances

Class	Precision	Recall	F1-score	Support
BONUS	0.57	0.82	0.67	749
CHANGE_INFO	0.60	0.79	0.69	853

OTHER	0.92	0.76	0.83	6920
REVOCATION	0.67	0.84	0.74	587
STATUS_QUESTION	0.57	0.71	0.63	812
Weighted average	0.81	0.77	0.79	9923
Accuracy			0.77	9923

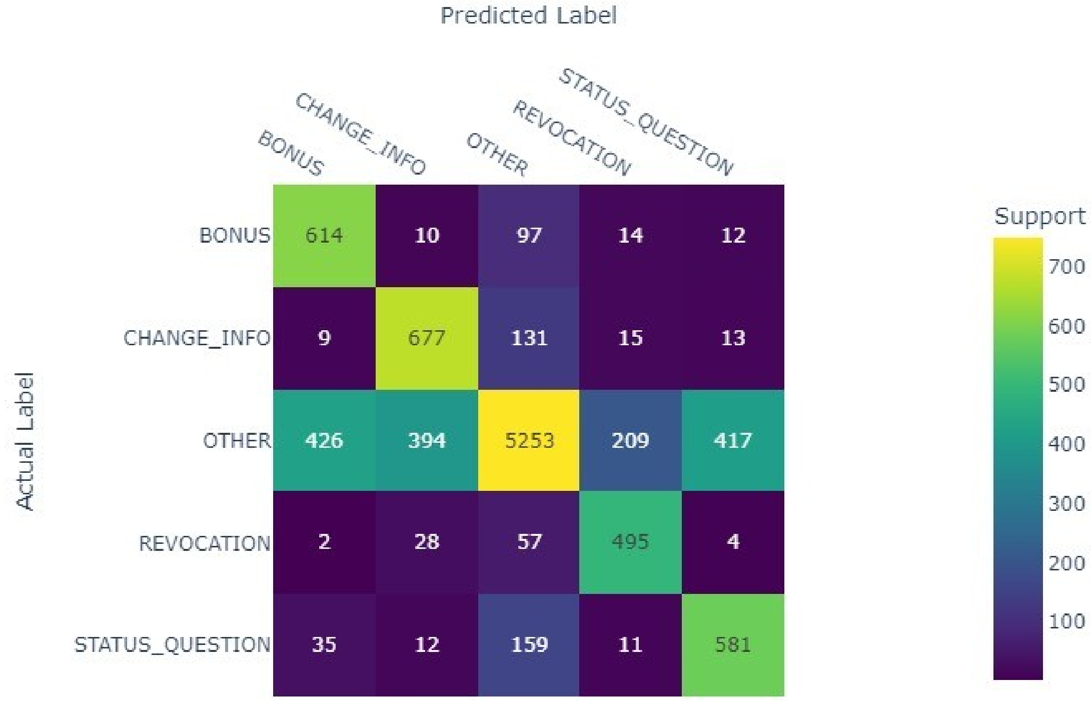


Figure 11: Confusion Matrix GNN 50000 data instances

The next trial excluded the class *OTHER* and involved approximately 30000 trainings and 7500 test instances. The results indicated an accuracy of 0.86 and a weighted average F1 score of 0.87. The F1 score of the relevant classes ranged from 0.85 to 0.90. In contrast to other trials the confusion matrix (figure 12) shows a more equal spread of misclassifications between classes.

Table 8: Results GNN 45000 data instances without OTHER class

Class	Precision	Recall	F1-score	Support
BONUS	0.92	0.85	0.89	1844
CHANGE_INFO	0.87	0.86	0.87	1980
REVOCATION	0.87	0.92	0.90	1335
STATUS_QUESTION	0.86	0.84	0.85	1621
Weighted average	0.88	0.86	0.87	6924
Accuracy			0.86	6924

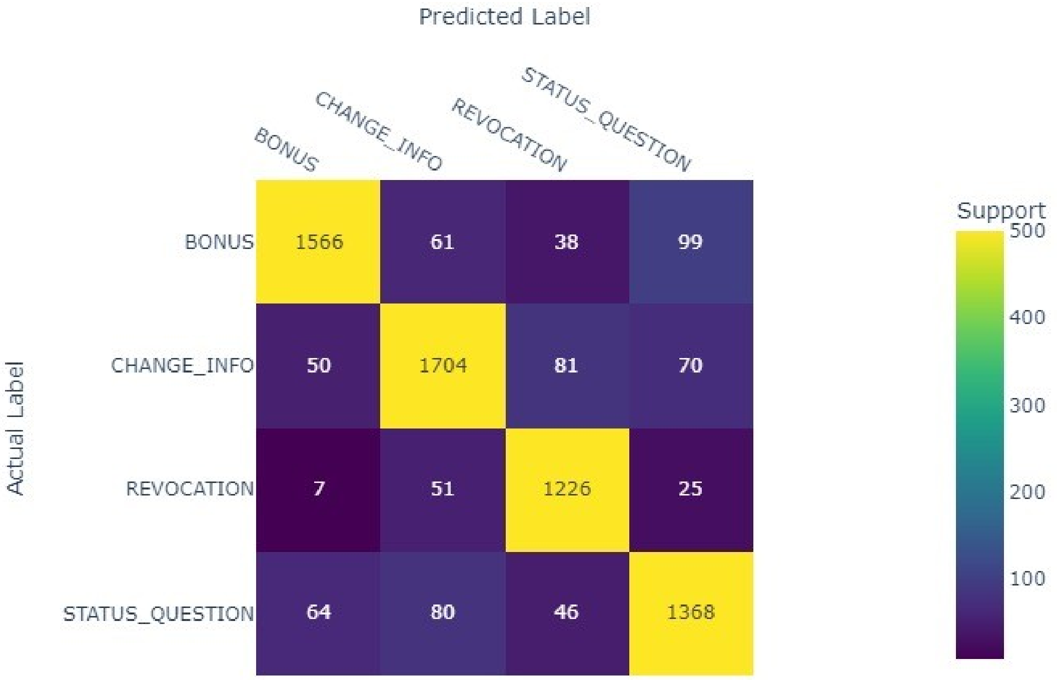


Figure 12: Confusion Matrix GNN 45000 data instances without OTHER class

The following trial used a reduced vector length of 50 and involved 20000 data instances. The findings showed an accuracy of 0.79 and a weighted average F1 score of 0.79. The F1 score of the relevant classes varied from 0.50 to 0.73. The confusion matrix (figure 13) turns out as expected.

Table 9: Results GNN 20000 data instances with a vector length of 50

Class	Precision	Recall	F1-score	Support
BONUS	0.52	0.83	0.63	254
CHANGE_INFO	0.42	0.63	0.50	183
OTHER	0.93	0.77	0.84	3122
REVOCATION	0.66	0.82	0.73	228

STATUS_QUESTION	0.37	0.74	0.49	175
Weighted average	0.83	0.77	0.79	4022
Accuracy			0.79	4022

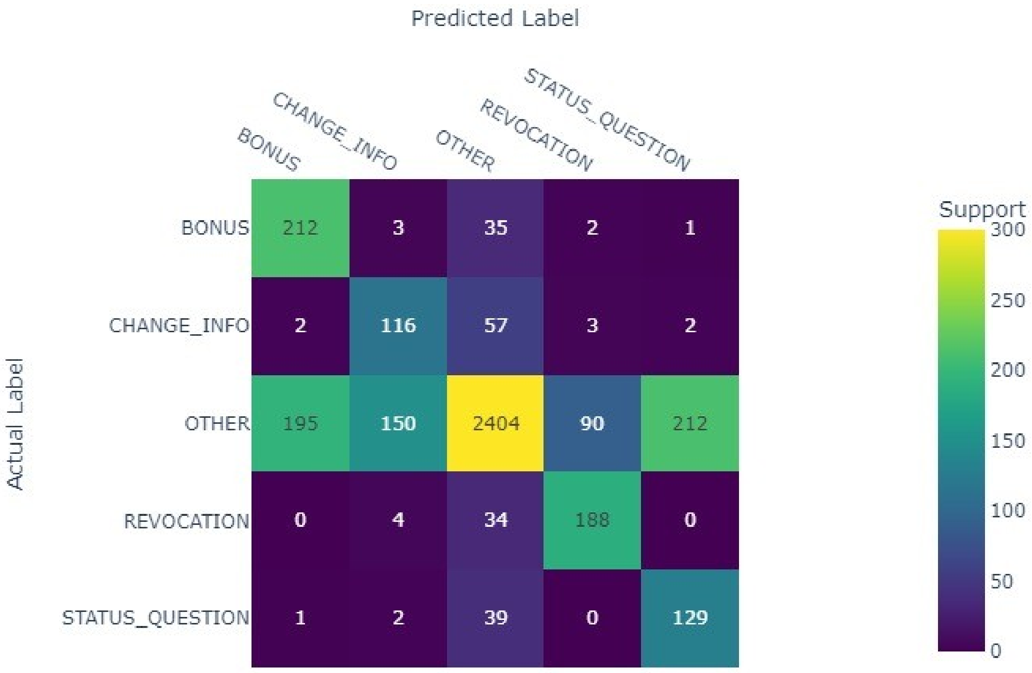


Figure 13: Confusion Matrix GNN 20000 data instances with a vector length of 50

In the next trial, the parameters were adjusted to increase the range of graph edges to 2. The results of this trial showed an accuracy of 0.79, a weighted average F1 score of 0.79, and F1 scores for the relevant classes ranging from 0.60 to 0.79. Except a higher portion of as OTHER misclassified instances compared to the previous trial, the confusion matrix (figure 14) yields no additional information.

Table 10: Results GNN 20000 data instances with range of edges = 2

Class	Precision	Recall	F1-score	Support
BONUS	0.64	0.78	0.70	312
CHANGE_INFO	0.57	0.64	0.60	276
OTHER	0.89	0.82	0.85	2827
REVOCATION	0.76	0.83	0.79	270
STATUS_QUESTION	0.57	0.68	0.62	309
Weighted average	0.80	0.79	0.79	4079
Accuracy			0.79	4079

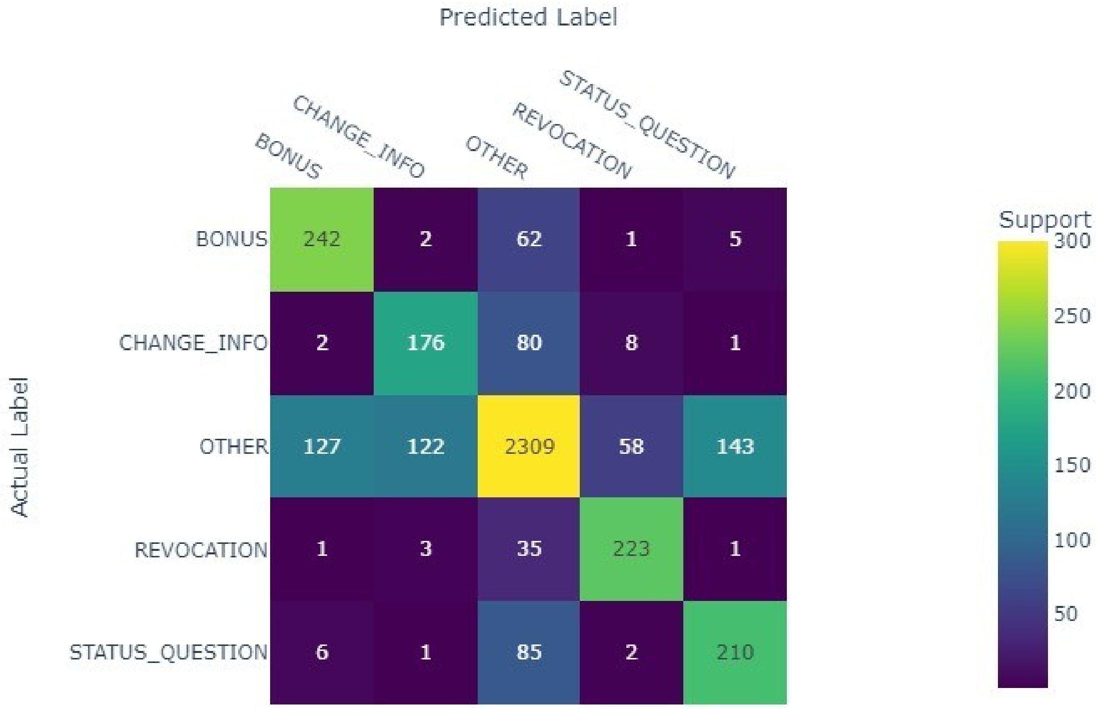


Figure 14: Confusion Matrix GNN 20000 data instances with range of edges = 2

The next trial will be increasing the number of graph layers to 4 and using 20000 data instances. The results of this trial yielded an accuracy of 0.80 and a weighted average F1 score of 0.80. The F1 score for the relevant classes were between 0.57 and 0.77

Table 10: Results GNN 20000 data instances with 4 graph layers

Class	Precision	Recall	F1-score	Support
BONUS	0.53	0.82	0.64	238
CHANGE_INFO	0.46	0.75	0.57	177
OTHER	0.93	0.78	0.85	2958
REVOCATION	0.75	0.80	0.77	289
STATUS_QUESTION	0.53	0.81	0.64	250

Weighted average	0.84	0.79	0.80	3949
Accuracy			0.80	3949

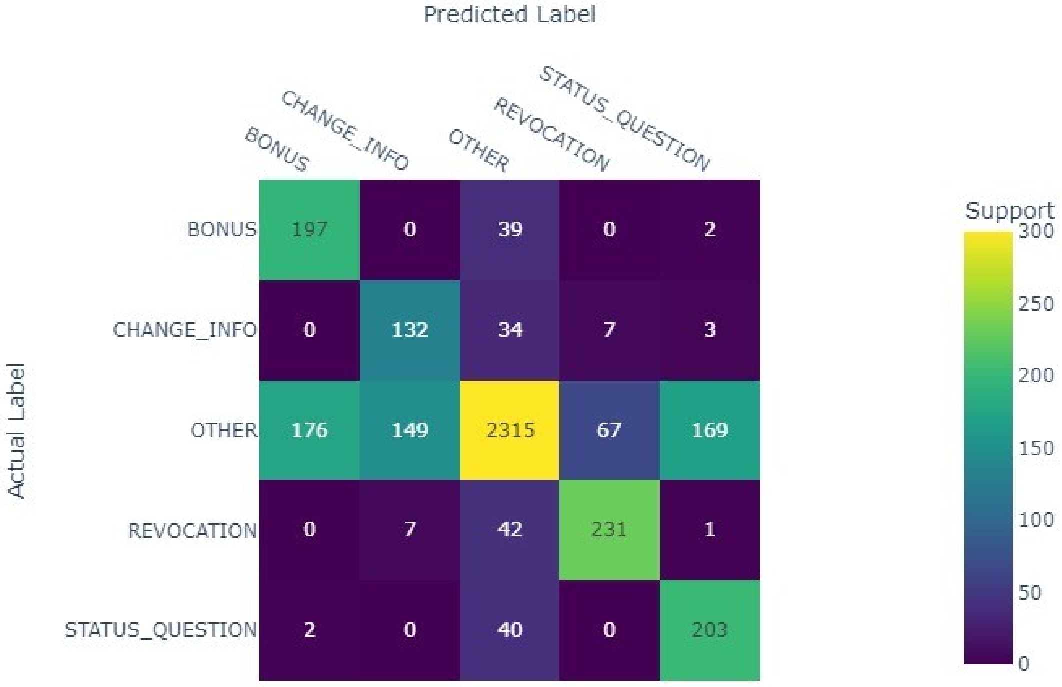


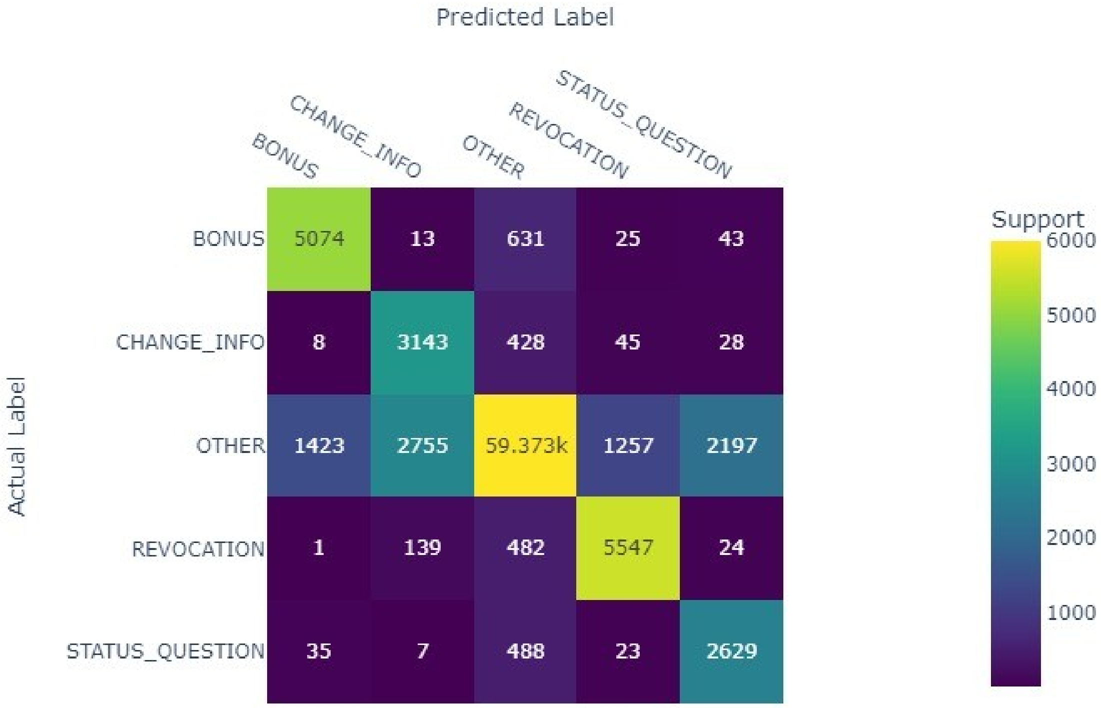
Figure 15 Confusion Matrix GNN 20000 data instances with 4 graph layers

5.4 Feed Forward Networks

In the subsequent trial, the Feed Forward Neural Network is utilized. The results of this trial yielded an accuracy of 0.88 and a weighted average F1 score of 0.89. The F1 score for the relevant classes ranged from 0.65 to 0.84. According to the confusion matrix (figure 16), similar to previous trials, a lot of *OTHER* messages are falsely classified in relevant classes. This observation however is less pronounced for the classes *REVOCATION* and *BONUS* in this trial.

Table 12: Results feed forward network 430000 data instances

Class	Precision	Recall	F1-score	Support
BONUS	0.77	0.88	0.82	5709
CHANGE_INFO	0.52	0.86	0.65	3656
OTHER	0.97	0.88	0.92	67540
REVOCATION	0.80	0.89	0.84	6163
STATUS_QUESTION	0.56	0.80	0.66	3458
Weighted average	0.91	0.88	0.89	86526
Accuracy			0.88	86526



	BONUS	CHANGE_INFO	OTHER	REVOCATION	STATUS_QUESTION
BONUS	5074	13	631	25	43
CHANGE_INFO	8	3143	428	45	28
OTHER	1423	2755	59.373k	1257	2197
REVOCATION	1	139	482	5547	24
STATUS_QUESTION	35	7	488	23	2629

Figure 16: Confusion matrix Feed Forward Network

In the last experiment all Data except the *OTHER* class are considered. This test found that both the accuracy and the weighted average F1 score amounted to 0.90.

The F1 scores for the relevant classes spanned from 0.89 to 0.94. The confusion between CHANGE_INFO and REVOCATION was notably higher than between other classes. This observation however is not as strong in this trial.

Table 13: Results feed forward network OTHER class excluded

Class	Precision	Recall	F1-score	Support
BONUS	0.93	0.95	0.94	6375
CHANGE_INFO	0.90	0.89	0.89	6037
REVOCATION	0.94	0.88	0.91	7383
STATUS_QUES	0.89	0.90	0.90	4957

TION				
Weighted average	0.91	0.90	0.90	25085
Accuracy			0.90	25085

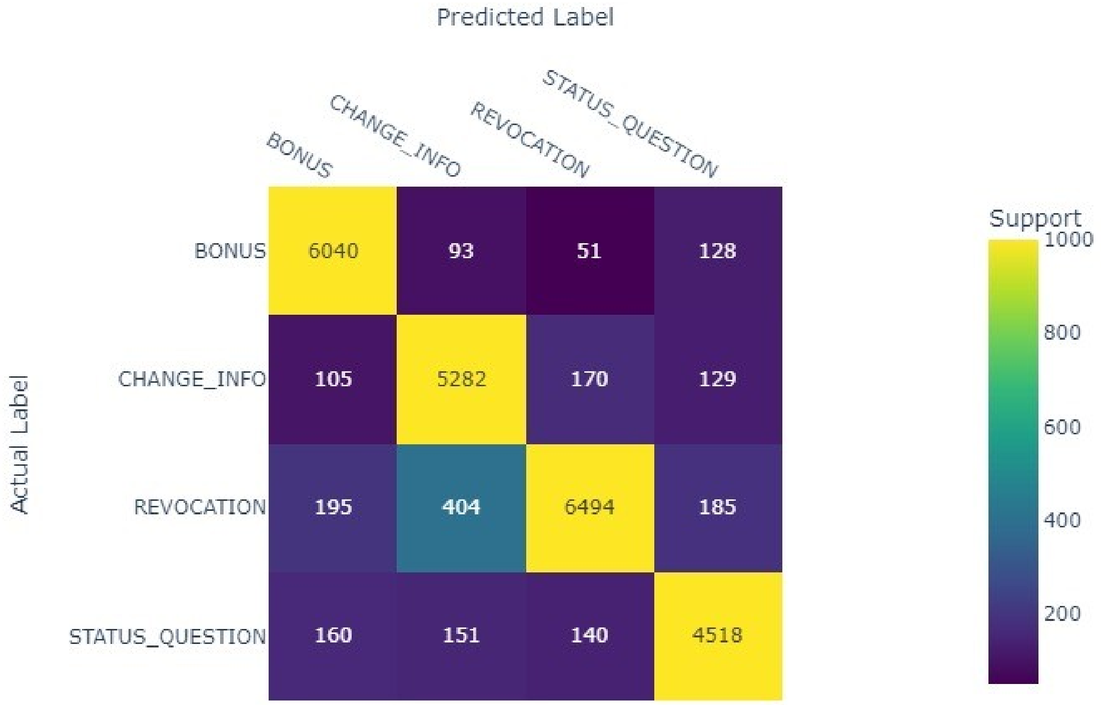


Figure 17: Confusion matrix Feed Forward Network excluding OTHER

5.5 Comparison

In this section the approaches are compared with each other. Considering the trials including the *OTHER* class (figure 18), the Feed Forward Neural Network achieves, with a F1-score of 0.89 and an accuracy of 0.88, the best results. Random Forest and Support Vector Machine score with a F1 score of 0.87 and an accuracy of 0.85 slightly worse. The Graph Neural Networks performance was with a F1 score of 0.80 and an accuracy of 0.80 notably worse.

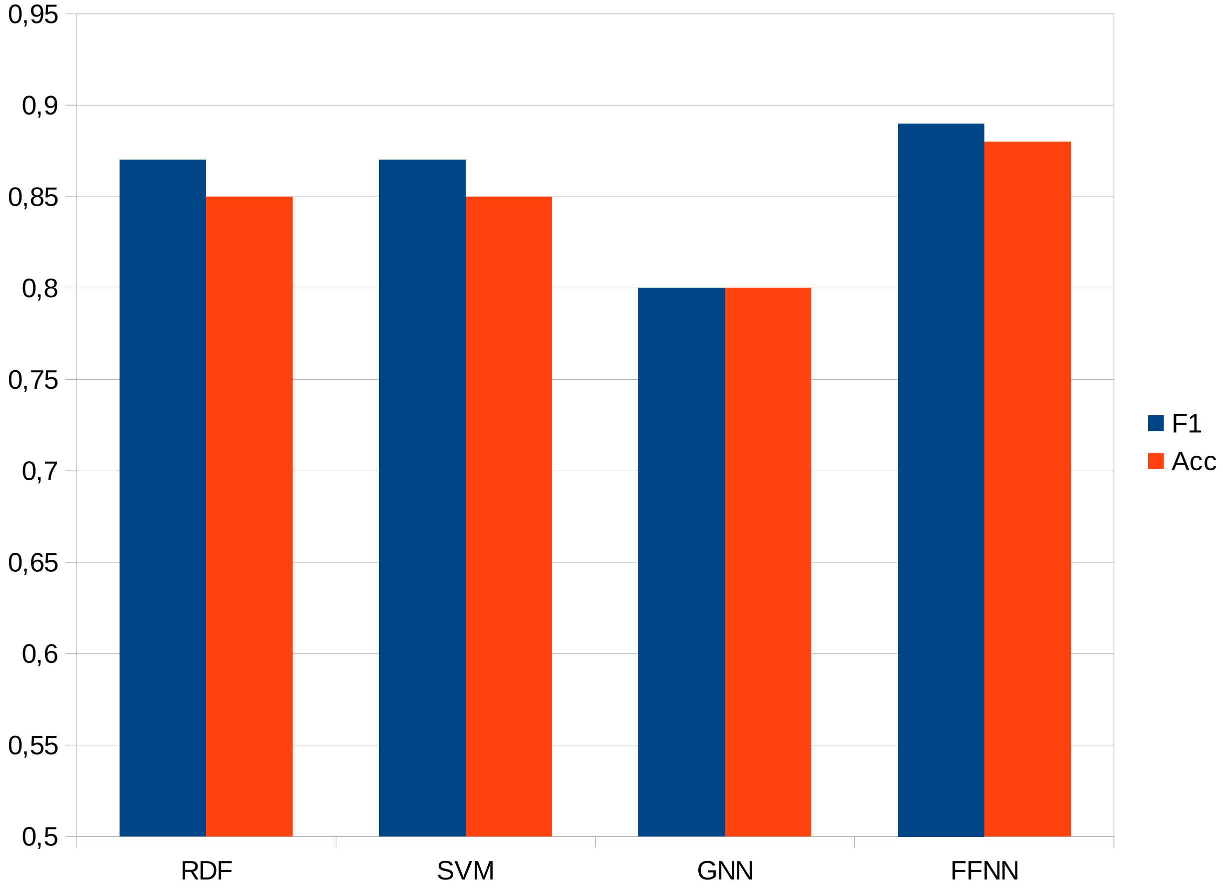


Figure 18: Best values of every approach including OTHER

In Figure 19 the detection of relevant classes in trials including the class *OTHER* is compared. For Graph Neural Networks the trial featuring an increased number of data instances (table 7) is considered, since it yielded the best results in that regard. The random forest algorithm achieved a weighted average F1 score of 0.69, while the Support Vector Machine had an F1 score of 0.72. The graph neural network had a slightly worse performance with an F1 score of 0.68. The feed forward neural network performed the best out of the four algorithms, with an F1 score of 0.76. Although the Graph Neural Network performed distinctively worse in general classification, its results connect to the other algorithms in detecting relevant classes.

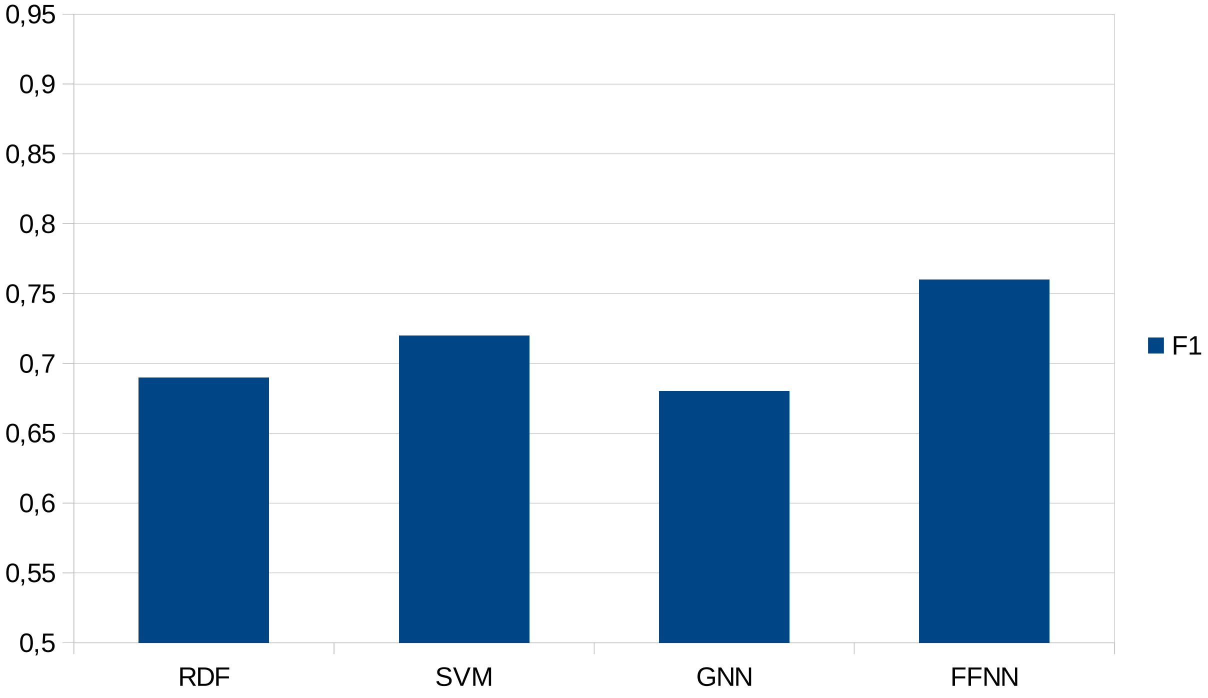


Figure 19: Weighted average of F1-scores of relevant classes in trials including OTHER

When comparing the trials excluding the *OTHER* class, the Feed Forward Neural Network performed the best, with a F1-score and an accuracy of 0.90. The Random Forest and Support Vector Machine scored slightly lower, with a F1-score of 0.89. The Graph Neural Network had again a slightly worse performance, with a F1-score of 0.87 and an accuracy of 0.86. Compared with Trials including the *OTHER* class (Figure 20), the overall performance increased for all algorithms. This improvement was expected, since by excluding the *OTHER* class, the differences between classes become better defined. Therefore the data is easier to classify.

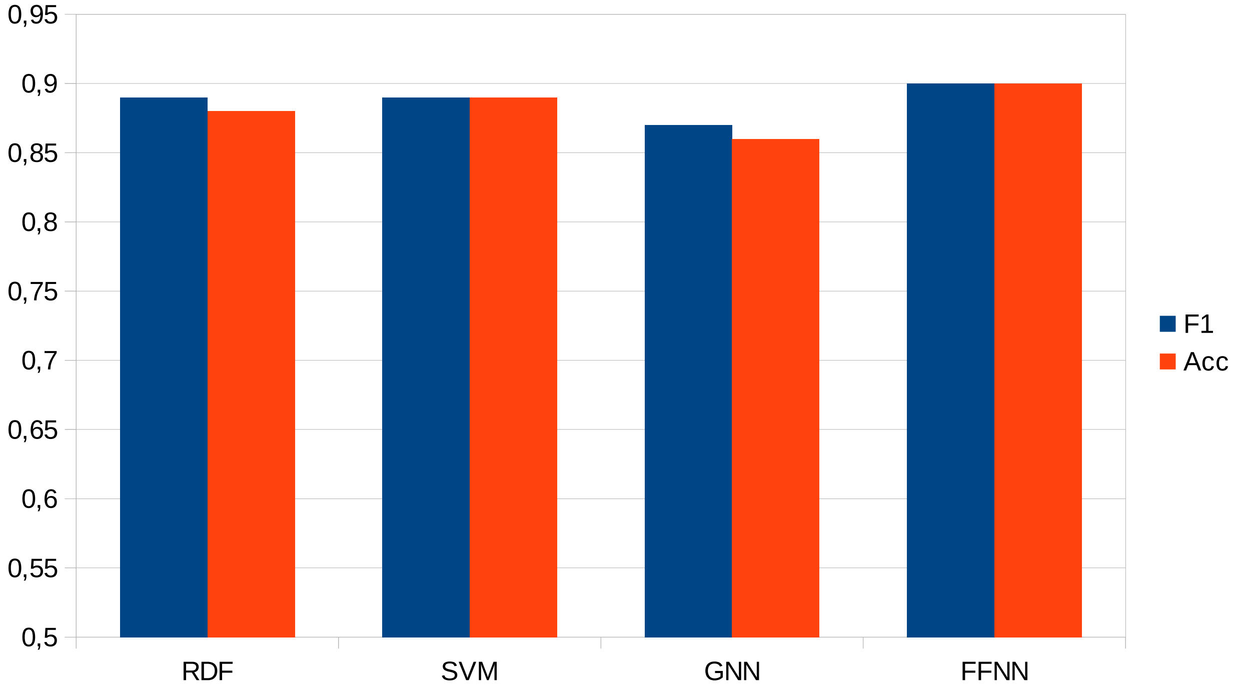


Figure 20: Results of Trials excluding OTHER class

Optimization attempts using Graph Neural Networks have only resulted in marginal improvement in performance. However, the addition of two more graph convolution layers has had the greatest impact on accuracy. Increasing the amount of training data resulted in the best improved detection of relevant classes. Despite these efforts, the overall performance of the Graph Neural Networks model is still notably worse compared to the other algorithms.

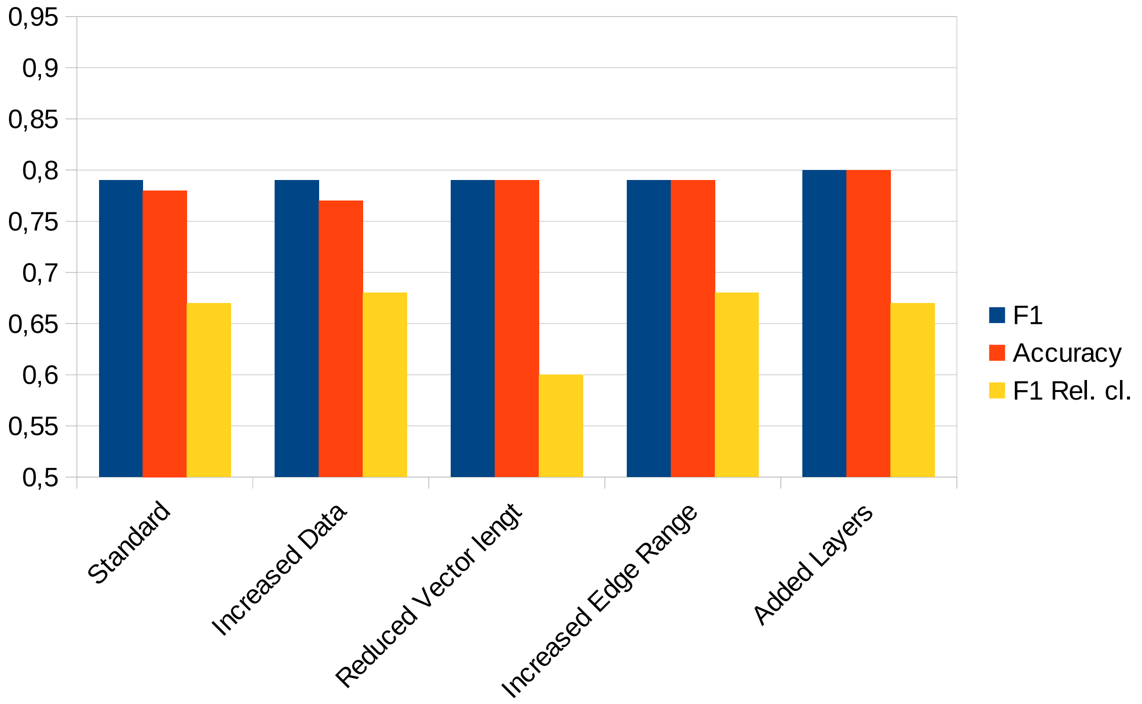


Figure 21: Results of GNN Parameter Optimization

6 Discussion

As far as vectorization is concerned, the Word2vec algorithm was applied successfully. The Tf-idf algorithms implementation resulted in excessively long vectors that required too much memory, making it impractical to be used in this work. Hence only data processed by Word2vec was classified. During the classification no issues regarding the Word2vec embeddings arose. Therefore Word2vec can be considered a fitting vectorization algorithm in this specific and similar problems.

Regarding the classification, the best performance was achieved by the Feed Forward Neural Network, while the Random Forest and Support Vector Machine showed slightly inferior performance. The Graph Neural Network however under performed compared to the other evaluated algorithms. This suggests that the text graph and the convolution layers processing it, do not only not improve, but hinder the classification. It is possible that the additional information contained within the text graph could lead to confusion for the model, thus causing a reduction in performance. Derived from this observation, the context between the words may only carry marginal information indicating the class of an e-mail, compared to the words themselves. Therefore Graph Neural Networks process a lot of meaningless information.

Furthermore, it is worth noting that the use of a text graphs significantly increases the computational cost of the classification task. This is an important consideration, as it may not be practical to use a Graph Neural Network in situations where computational resources are limited. The smaller amount of training data, that was available to the Graph Neural Network owed to said greater computation cost, could also explain the algorithm's underperformance. However, the fact that increasing the available data from 20000 to 50000 instances did not yield significant improvement does not support this explanation. The other classification algorithms had significantly lower computation cost. The duration of the calculation for Random Forest, Support Vector Machine and Feed Forward Neural Network are comparable with the latter being the shortest (table 2).

It is also important to note that the conclusions of this study are limited by data only from a specific domain. While the results of this study may be meaningful in the context of this data set, it is unclear how well they would generalize to other data sets or other real-world scenarios.

To return to the goal of this work, the optimal workflow to solve the said classification problem was to be determined. Concerning the vectorization, the Word2vec algorithm yielded a working base for the classification. For classification the Feed Forward Neural Network appeared to be the optimal choice. This algorithm yielded the best classification results while maintaining low computational cost.

Therefore according to this work, Word2vec for vectorization and the Feed Forward Neural Network solve this classification problem best.

Bibliography

Bahi, H.A.A., 2021. Master Thesis Optimization of hyperparameters of ANNs – Application to the second Virial coefficient (B) of fluid mixtures.

Boser, B.E., Guyon, I.M., Vapnik, V.N., 1992. A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92. Association for Computing Machinery, New York, NY, USA, pp. 144–152. https://doi.org/10.1145/130385.130401

Bouaziz, A., Dartigues-Pallez, C., da Costa Pereira, C., Precioso, F., Lloret, P., 2014. Short Text Classification Using Semantic Random Forest. https://doi.org/10.1007/978-3-319-10160-6_26

Breiman, L., 2001. Random Forests. Machine Learning 45, 5–32. https://doi.org/10.1023/A:1010933404324

Bridle, J.S., 1990. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition, in: Soulié, F.F., Hérault, J. (Eds.), Neurocomputing, NATO ASI Series. Springer, Berlin, Heidelberg, pp. 227–236. https://doi.org/10.1007/978-3-642-76153-9_28

Crammer, K., Singer, Y., 2001. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines.

CSIRO's Data61, 2018. StellarGraph Machine Learning Library. GitHub Repository.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning, Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts ; London, England.

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci U S A 79, 2554–2558.

Huang, L., Ma, D., Li, S., Zhang, X., WANG, H., 2019. Text Level Graph Neural Network for Text Classification.

Ikonomakis, M., Kotsiantis, S., Tampakas, V., 2005. Text Classification Using Machine Learning Techniques.

Jacovi, A., Shalom, O.S., Goldberg, Y., 2020. Understanding Convolutional Neural Networks for Text Classification.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An introduction to statistical learning with applications in R. Statistical Theory and Related Fields. https://doi.org/10.1080/24754269.2021.1980261

Kim, H., Howland, P., Park, H., 2005. Dimension Reduction in Text Classification with Support Vector Machines 17.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D., 2019. Text Classification Algorithms: A Survey.

Lever, J., Krzywinski, M., Altman, N., 2016. Classification evaluation. Nat Methods 13, 603–604. https://doi.org/10.1038/nmeth.3945

Liang, Y., Chengsheng, M., Yuan, L., 2019. Graph Convolutional Networks for Text Classification. The Thirty-Third AAAI Conference on Artificial Intelligence 7370–7377.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/arXiv.1301.3781

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J., 2021. Deep Learning Based Text Classification: A Comprehensive Review.

models.word2vec – Word2vec embeddings — gensim [WWW Document], n.d. URL https://radimrehurek.com/gensim/models/word2vec.html (accessed 1.10.23).

Nair, V., Hinton, G.E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines.

NLTK :: nltk.stem.snowball module [WWW Document], n.d. URL https://www.nltk.org/api/nltk.stem.snowball.html (accessed 2.6.23).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., n.d. Scikit-learn: Machine Learning in Python. MACHINE LEARNING IN PYTHON.

Pranckevičius, T., Marcinkevičius, V., 2017. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. BJMC 5. https://doi.org/10.22364/bjmc.2017.5.2.05

Qing, L., Linhong, W., Xuehai, D., 2019. A Novel Neural Network-Based Method for Medical Text Classification. Future Internet 11, 255. https://doi.org/10.3390/fi11120255

Rosasco, L., Vito, E.D., Caponnetto, A., Piana, M., Verri, A., 2003. Are Loss Functions All the Same? Neural Computation 16, 1063–1076. https://doi.org/10.1162/089976604773135104

Sánchez-Pi, N., Martí, L., Garcia, A.C., 2014. Text Classification Techniques in Oil Industry Applications. pp. 211–220. https://doi.org/10.1007/978-3-319-01854-6_22

Scarselli, F., Gori, M., Ah Chung Tsoi, Hagenbuchner, M., Monfardini, G., 2009. The Graph Neural Network Model. IEEE Trans. Neural Netw. 20, 61–80. https://doi.org/10.1109/TNN.2008.2005605

sklearn.ensemble.RandomForestClassifier — scikit-learn 1.2.0 documentation [WWW Document], n.d. URL https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (accessed 1.12.23).

sklearn.feature_extraction.text.TfidfVectorizer [WWW Document], n.d. . scikit-learn. URL https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (accessed 4.3.23).sklearn.metrics.classification_report [WWW Document], n.d. . scikit-learn. URL https://scikit-learn/stable/modules/generated/sklearn.metrics.classification_report.html (accessed 1.27.23).

sklearn.model_selection.HalvingGridSearchCV [WWW Document], n.d. . scikit-learn. URL https://scikit-learn/stable/modules/generated/sklearn.model_selection.HalvingGridSearchCV.html (accessed 1.12.23).

sklearn.svm.LinearSVC [WWW Document], n.d. . scikit-learn. URL https://scikit-learn/stable/modules/generated/sklearn.svm.LinearSVC.html (accessed 1.13.23).

Sparck Jones, K., 1972. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28, 11–21. https://doi.org/10.1108/eb026526

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting.

Sun, X., Lu, W., 2020. Understanding Attention for Text Classification, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Presented at the ACL 2020, Association for Computational Linguistics, Online, pp. 3418–3428. https://doi.org/10.18653/v1/2020.acl-main.312

Vapnik, V.N., Chervonenkis, A.Ya., 1964. A class of algorithms for pattern recognition learning.

Verma, P., Goyal, A., Gigras, Y., 2020. Email phishing: text classification using natural language processing. Comput. Sci. Inf. Technol. 1, 1–12. https://doi.org/10.11591/csit.v1i1.p1-12

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2021. A Comprehensive Survey on Graph Neural Networks. IEEE Trans. Neural Netw. Learning Syst. 32, 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Zhang, M., Cui, Z., Neumann, M., Chen, Y., 2018. An End-to-End Deep Learning Architecture for Graph Classification. Proceedings of the AAAI Conference on Artificial Intelligence 32. https://doi.org/10.1609/aaai.v32i1.11782