THE HAGUE
UNIVERSITY OF
APPLIED SCIENCES

Atos

Ticket to Category

text classification on technical support tickets

Niels van Beathem

Approved: .....
R. Hoogsteen

Bachelor thesis· 30 credits
The Hague University of Applied Sciences, HHS
Technic, Innovation & Society
Atos
Delft, 2021

Colophon

Document	:	bachelor thesis
Version	:	4
Date	:	December 21, 2021
Font	:	Computer Modern Roman, 10
Cover	:	image by belterz

Name student	:	Niels van Benthem
Number student	:	14113899
Mail student	:	14113899@student.hhs.nl

Company	:	Atos
Adress	:	Burgemeester Rijnderslaan 30, 1185 MC Amstelveen
Name supervisor 1	:	Roeland Hoogstraten
Mail supervisor 1	:	roeland.hoogstraten@atos.net
Name supervisor 2	:	Urscha Fajdiga
Mail supervisor 2	:	urscha.fajdiga@atos.net

University	:	The Hague University of Applied Sciences
Adress	:	Rotterdamseweg 137, 2628 AL Delft
Bachelor	:	mathematical applications
Name mentor	:	Jeroen B.P. Vuurens
Mail mentor	:	j.b.p.vuurens@hhs.nl

Preface

Focusing on the improvement of the end user experience is a new approach, in an industry where reacting afterwards is the standard. This consequently results in the opposite effect to what is desired, improving the employee experience of customers. An idea is pitched to offset this behaviour and to be proactive in solving the end user's issue before it occurs. By analysing the existing technical support tickets, I attempt to map the most common problems, creating the opportunity for Atos to give a solution, preventing a ticket of the same issue to be made by a new end-user. An essential part in analysing tickets is classifying the category correctly, since wrong categorisation could lead to improper advice.

This paper encompasses a modern approach in the technical support branch and touches upon the subject of natural language processing, specifically text classification. I study the effect of categorising tickets with machine learning using the descriptive data of the tickets. This is of interest for data scientists working in customer services.

Chapter 1 provides a short introduction in describing the context of the environment and organisation, and who will use the created model. In case you are interested in the business value, please read Chapter 2. Chapter 3 states the scientific ground on the subject of text classification up to the publishing of this study. The readers who are curious in the steps towards creating a model can read Chapter 4 and 5. In Chapter 6 and 7, you can examine the results including a description of the usability. Lastly, for what this research yields and recommends, please read Chapter 8 and 9.

I would like to give my appreciation to my supervisors, Urscha Fajdiga and Roeland Hoogstraten from Atos, the client Atos Prasenjit Bose, the Atos PXC Developments team and The Hague University of applied sciences, especially, Dr. Jeroen B.P. Vuurens, for their guidance and unwavering support.

Delft, December 2021 Niels van Benthem

Abstract

The Proactive Experience Center is a practice within Atos, that focuses on proactive workplace management to measure the employee experience of customers. They do this for the following reasons:

- Improving the digital workplace;
- Creating digital inclusion;
- Caring for employee well-being;
- Enhancing customer satisfaction;
- Increasing business outcome.

To achieve this, employee experience needs to be measured. One data source for this, is the technical support ticket of the employee from a customer. This provides insight into which problems need to be addressed, to improve the employee experience. However, currently, the analysis of the ticket data is done on a small portion of ticket data. This is because the category, which is essential in providing the context, is lacking or otherwise often inaccurate and it is impractical for an agent to correctly categorise every ticket for analysis. That means that a solution is needed that can effectively categorise tickets based on the written text. To solve this problem, the main question is formulated as follows:

> *'How effective is a text classification model in classifying the category of technical support tickets, based on descriptive data provided by tickets?'*

First of all, to reduce the complexity and limit the number of models, the scope is on English written tickets and three models, where at least one model is from the domain machine learning.

Secondly, background information is gathered to get insight in why this is important and how to approach this problem. At the same time, the possible data sets that Atos has to offer are inspected. This provided the necessary information to keep the focus of the research on predicting the category of tickets that were written by the end-user sent to a support team.

Literature provided three models; (1) support vector machines, (2) deep neural network, and (3) recurrent neural network.

The text preprocessing is identical for each model. Where it differs is the feature engineering. To compare them, deep neural network uses both word representations. Support vector machines and deep neural network use term frequency-inverse document frequency. Recurrent neural network and another deep neural network use sequences

in combination with an embedding layer.

To validate the research, the hyperparameters are tuned with cross-validation, and the effectiveness of the model is scored based on the weighted F_1 score, which takes class imbalance into account.

This concluded that recurrent neural networks are the best in classifying the category of tickets. The model scored a weighted F_1 score of 85%. However, the results indicated that overlapping categories and/or class imbalance affect the model in predicting the correct label. The comparison between the deep neural networks revealed that retaining context and semantics is beneficial, indicating that word embedding is the way forward.

The research exposed problems with the data and provided a solution for them. However, certain topics are not addressed. The following tasks are recommended to address the problems and topics:

- Improve the definition of the categories;
- Keep the non-English written tickets;
- Measure the individual steps to calculate the business value;
- Assess the effect of out-of-vocabulary words;
- Research models that are similar in complexity as recurrent neural networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motive

Atos is organised in so called practices. Every practice focuses on a specific part of the IT-services Atos delivers to its customers. One of these practices is the Digital Workplace practice focusing on end-user support and workplace services with a major focus on Employee Experience (hereinafter EX). One of the services the Digital Workplace practice delivers is the Proactive Experience Center (hereinafter PXC). The PXC's goal is to increase the capabilities for proactive workplace management instead of the traditional reactive workplace management. As Atos believes being proactive will increase the EX, which is the ultimate goal of the PXC. One of the inputs the PXC uses to achieve the aforementioned, is to analyse customer's ticket data. Atos believes that providing feedback on how to resolve the most prevalent problems will increase its capabilities to be more proactive in resolving issues, perhaps even before the employee themselves is aware of the issue.

The analysis of the ticket data is done manually, thereby, the quality of the analysis is highly dependent on the domain knowledge and expertise of the agent performing the analysis. A guide for the agent is the category field, since it narrows the possible subjects the ticket relates to, however, it is often wrongly assigned or ambiguous. Not having a reliable indication results in classifying errors and longer process time per ticket. Since the agent has no indication of the category, more time is spent on identifying the problem. The classifying error is due to the complexity of the ticket. The more complex the ticket is, the more likely it is that an allocation error occurs, due to the fact that a subject can be based on the wrong problem.

The ultimate goal of PXC is achieved better with higher quality data analysis, because the data analysis leads to giving feedback to the customer about what needs to be improved or changed. The quality of the data analysis is based on whether the most prevalent problem has been found. If it is not found, the improvements that are reported back to the customer are incomplete, resulting in the most prevalent problem continuing to affect new end-users, negatively affecting the EX. Therefore, simplifying the data analysis is important, which can be done by reducing the set of possible problems that can be assigned to a ticket. This set is based on the category of the ticket e.g., hardware issues cannot have a problem related to MS Outlook, making it easier to determine the

issue by the agent. Hence, it is essential to have reliable classification of the category. In addition, the agent has the possibility to filter out categories that cannot be resolved by the customer, as a result freeing up time for the agent to spend on other projects.

1.2 Objective

The intent of the graduation thesis is twofold:

(1) researching the best way to classify tickets, in the domain of natural language processing (hereinafter NLP);
(2) determining if the chosen method can sufficiently assign categories.

After researching and designing a suitable model, the effectiveness of the model is assessed using the predicted and actual categories, by means of the weighted F_1 score. To do that the next research question is drafted:

> *'How effective is a text classification model in classifying the category of tickets, based on descriptive data provided by tickets?'*

Answering this question is done with the aid of three sub-questions:

(1) *'Which models are used for classifying tickets?'*
(2) *'What is the most suitable way to preprocess ticket data?'*
(3) *'Which model should Atos use, in the given scenario, to classify tickets?'*

The research is done on the basis of a literature review, knowledge transfer by domain experts (e.g., thesis supervisors) and desk research.

The tickets are made up of different natural languages, adding the domain of neural machine translation. Since Atos' corporate language is English, only English tickets are processed to avoid the need for translation. In addition, the effectiveness of the model is tested on a portion of the ticket data. Lastly, a list of approved categories is created, in consultation with domain experts, as some categories are customer-specific or ambiguous.

We are only considering three models involving both deep learning and machine learning, where at least one model is from the domain machine learning.

1.3 Reading Guide

This report consists of nine chapters describing the issue at hand, providing an answer to the research question, or providing insight in potential future endeavours. Chapter 2 and 3 describe the business value, problems and previous work in the text classification field. Insight is procured in Chapter 4 by analysing the ticket data and the problem context. The knowledge gained is applied in Chapter 5 for ticket data preparation and empirical study. The results of the study are conveyed in Chapter 6. Chapter 7 interprets the choices that were made, the results, the implications it had and how to overcome them. Conclusions on the best suited text classification model for classifying technical support tickets are described in Chapter 8. In Chapter 9, the potential steps are described for what can be done in future research.

Chapter 2

Background

2.1 Move from Serve to Care

Atos has a lot of IT-services they provide to their customers, as described in Chapter 1. Before PXC was established, four years ago, these services were entered into a Service Level Agreement (hereinafter SLA). However, the problem with SLA is, that on paper Atos is providing what was agreed to, but the EX of the customer can be different, experiencing for example, connection errors, permission problems, bugs, etc.. In the industry this is known as the 'watermelon effect' (Ellis & Ellison, 2015), where metrics appear 'green', suggesting that everything is under control, yet, digging below the surface will readily reveal signs of 'red', indicating signs of ill-health. Atos, is tackling this by focusing on EX, offering customers Experience Level Agreements (hereinafter XLA). XLA differs from SLA in moving from only providing a service, to additionally, caring for the employees. With the introduction of the PXC, Atos attempts to come up with methods to measure and increase the EX. This project is focused on one method to accomplish this. By creating a reliable categorisation for the tickets, the PXC anticipates the following three effects:

(1) they can perform analysis of the category based on the entirety of the ticket data instead of a sample. The sample would have been manually labelled by an agent;
(2) the findings on what the problems are in specific categories can be analysed. PXC can designate agents to specific categories to assess the tickets more granular;
(3) continuing on the previous effect, less errors are made during this process since the agent has a better understanding on what the ticket is about, considering the category is provided.

2.2 Tickets

Atos collects ticket data from three sources:

(1) chatbot: descriptive data containing two participants, one is the end-user and the other is a computer program that simulates human conversation;
(2) IT service management tooling: descriptive data with the description provided by the end-user and the work notes by the support team;
(3) conversations: audio data of an end-user and support team conversation.

This project focuses on source 2, IT service management tooling, tickets that were written and labelled. Labelled data reduces the complexity of the problem by providing the correct answer, that can then be fed back to the model. With these tickets, the PXC seeks to inform customers on improvements, e.g., give a seminar for Azure. These improvements stem from the data analysis performed by an agent, who manually assigns the root cause of the ticket out of a predetermined list. There is a chance that the agent assigns the wrong problem, this has a few causes:

(1) the ticket is extensive, and causes the agent to focus on the wrong topic;
(2) miss-clicking when assigning the type;
(3) overlooking a remediable ticket, caused by an abundance of useless tickets;
(4) wrong categorisation of the ticket resulting in an incorrect type.

These result in an incorrect analysis of the data, which in return lead to different improvements advised to the customer.

A method to minimise those incidents is to provide a reliable category. Often the customers have their own set of labels, where the majority is grouped under a meaningless category, or a portion is posted in the wrong class. As a consequence, the agent cannot rely on the category, which provides essential information regarding the topic.

Chapter 3

Related Work

This report focuses on investigating the effectiveness of using text classification on predicting the category of tickets. This chapter gives an overview of the existing literature studies on the problem of data preparation, feature engineering and text classification. Afterwards, a summary is provided of the substantiations for the different subjects that support the goal of measuring the effectiveness of text classification on ticket data.

3.1 Data preparation

This section is split into two problems; data imbalance and text preprocessing. These two are chosen since, the exploratory data analysis showed the imbalanced nature of tickets. This could have an effect on the effectiveness of the model, and the data consists of written text. Research on these will show if it is necessary to balance and clean the data, and the techniques to do it.

3.1.1 Data imbalance

Leevy, Khoshgoftaar, Bauder, and Seliya (2018) describe that there are four categories and various techniques to deal with class imbalance:

(1) data-sampling:
  (1)(1) over-sampling;
  (1)(2) under-sampling;
(2) feature-selection;
(3) cost-sensitive;
(4) and hybrid/ensemble.

Liu (2004) researches, seventeen over- and under-sampling techniques, on imbalanced descriptive data for text classification and states that it is dataset dependent. Wong, Gatt, Stamatescu, and McDonnell (2016) reinforce the use of data-space augmentation methods. They observe better performance compared to augmentation in feature-space. They conclude that data-sampling approaches, such as over-sampling and data-space compression like under-sampling, are more suitable. However, this does not guarantee an improvement in classifying, seeing that sometimes it is reported to be ineffective

and may cause negative effect on multi-class tasks (Zhou & Liu, 2006; Poolsawad, Kambhampati, & Cleland, 2014; Yanminsun, Wong, & Kamel, 2011). The impact of class imbalance on machine learning and deep learning models is dependant on the size of the training set. Yanminsun et al. (2011) states that when the training set increases, the large error rate caused by the imbalanced class distribution decreases. This indicates that the need for data resample is unnecessary as long as the number of tickets is large enough. The large error rate is one of the effects of class imbalance, however, another effect is the bias of classification models. Wang and Zhang (2018); Akkaradamrongrat, Kachamas, and Sinthupinyo (2019) state that imbalanced data leads to the bias of classification models. They used machine learning and neural networks with oversampling, to resample the minority class to solve that bias. But, Padurariu and Breaban (2019) show mixed results depending on the machine learning model and oversampling technique. Another view is given by Poolsawad et al. (2014), they state that a trade-off is made since imbalance is a factor of the data, and throwing away valuable information is always possible when resampling the data.

3.1.2 Text preprocessing

Vijayarani, Ilamathi, and Nithya (2015); Kowsari et al. (2019); Revina, Buza, and Meister (2020) provide an overview of common preprocessing techniques, such as stop-word removal, slang and abbreviation correction, text cleaning and stemming. In deep learning, Camacho-Collados and Pilevar (2017) conclude that common preprocessing steps, lowercasing, lemmatising, and multi word grouping, work equally or worse compared to simple tokenization. They observe domain-specific data as an exception. In their study, text preprocessing showed to increase the deep learning's performance. For machine learning, Leopold and Kindermann (2002) investigate the different weighting schemes for the representation of text in input space. They state that for Support Vector Machines (hereinafter SVM) removing rare words and stop-words was unnecessary.

3.2 Feature engineering

Bansal (2018); Kowsari et al. (2019); Revina et al. (2020) show the different representations of words for text analysis, such as bag-of-words, word embedding, and raw corpus. Bag-of-words is a document representation where the structure of the sentence is lost, only the number of words matter. The frequency of each word is used as a feature (Zhang, Jin, & Zhou, 2010). A version that incorporates information on the more important words and the less important ones is term frequency-inverse document frequency (hereinafter TFIDF) (Revina et al., 2020). Word embedding maintains the structure of the sentence and captures the semantic of words. Each word is represented as a vector, such that the words that are closer in the vector space are expected to be similar in meaning (Lai, Liu, He, & Zhao, 2016). Word embeddings need to be trained, but pre-trained models exist, such as Google's Word2Vec[1], Stanford's GloVe[2], Facebook's fastText[3] (Revina et al., 2020; Kowsari et al., 2019). However, Wahba, Madhavji, and Steinbacher (2020) researched the effectiveness of different pretrained word embeddings, including domain-specific word embedding on IT support tickets, and concluded that IT support tickets do not benefit from pretrained word embeddings.

[1]Word2Vec
[2]GloVe
[3]fastText

This is due to domain-specific words considered to be out-of-vocabulary[4] for pre-trained embeddings. Training a domain-specific word embedding has two major disadvantages according to Revina et al. (2020), (i) large quantity of data necessary, and (ii) only learns the words that appear in the data. However, on the other hand it is able to consider syntax, semantics and polysemy.

3.3 Text classification

Kowsari et al. (2019); Revina et al. (2020); Zemp (2021); Wahba et al. (2020) give an overview on architectures that can be used in text classification, such as recurrent neural networks (hereinafter RNN). Study of Lyubinets, Boiko, and Nicholas (2018) on classifying data from customer service systems used RNN with word embedding and outperformed the classic solutions for the task. While Han and Akbari (2018) achieved the best outcome with convolutional neural network (hereinafter ConvNet) for the task of classifying technical support tickets. The master thesis of Zemp (2021) examines the effectiveness of ConvNet, bi-directional long short-term memory (hereinafter BiLSTM), gated recurrent units and transformers on service desk tickets. All of them outperform machine learning approaches, like logistic regression and random forest. Minaee et al. (2021); Kowsari et al. (2017) state that deep neural networks (hereinafter DNN) are the simplest deep learning model. However, it has achieved high accuracy on many text classification benchmarks. Which does not mean that it is better than other deep learning architectures, but it can contest them in some instances.

3.4 Overview

I note the lack of a clear method and model that should be used in classifying tickets. This is something I expect, seeing that the majority of the articles address that it differs per use case. However, it provided tools to apply the data preparation, feature engineering and text classification. I note the current trend of using Bi-directional Encoder Representation from Transformers (BERT) architecture for solving text classification problems. Nonetheless, I have my doubts considering the average word count of the tickets and the result after combining three fields, erasing the structure of the sentence, which is explained in Chapter 4.

All of the aforementioned stages for establishing a model to classify tickets have a form of uncertainty, primarily if it should be applied or not. Since this research focuses on the effectiveness of text classification on tickets, certain variations that I expect to have a minor impact on the effectiveness, are excluded. An overview of these decisions are merged in Table 3.1.

[4]Out-of-vocabulary are terms that are not part of the normal lexicon found in a natural language processing environment.

Table 3.1: Per subject is described what the reasoning behind the decision is

Subject	Method	Reason	Literature
imbalance	resampling	Imbalanced classes can be a problem that affect the effectiveness of the model. Yet, its impact is decreased when the dataset is large enough. Since oversampling has some caveats of its own, such as overfitting, data-space or feature-space resampling, and different methods (ADASYN, SMOTE and synonym replacement), I decided not to balance the data considering the impact being solvable with more data.	(Yanminsun et al., 2011; Zhou & Liu, 2006; Wong et al., 2016)
preprocessing	lowercasing, stop-words, word-removal and lemmatisation	Text preprocessing, in some cases, is unnecessary. However, in domain-specific instances, deep learning benefits from it. In addition, it reduces the dimensionality which helps with the amount of allocated RAM required. These arguments convinced me to apply text preprocessing	(Camacho-Collados & Pilevar, 2017; Zemp, 2021)
	slang, abbreviation and spelling correction	These are advanced preprocessing steps, which require a combination of time and domain knowledge. Seeing that I lack domain knowledge, I chose to ignore these steps.	(Kowsari et al., 2019)
word representation	TFIDF and word embedding	A combination of these two is selected. The possibility exists that there is not enough data for word embedding. But, it can be better then TFIDF since it considers semantics. Pretrained word embeddings are ignored seeing that IT support tickets do not benefit from it. TFIDF applies a term weighting scheme which can be helpful to reflect how important a word is.	(Wahba et al., 2020; Zemp, 2021; Revina et al., 2020; Kowsari et al., 2019)
architecture	SVM, DNN, RNN	Three models are compared to keep it manageable. Although, ConvNet performed better than DNN and SVM, it is left out. One machine learning solution is desired to have a baseline for what a more complex architecture should reach. For the machine learning model, SVM is used, as it is less effected by class imbalance. The simplest feed forward network is chosen to measure the difference in effectiveness, switching to neural networks, specifically multilayer perceptron (hereinafter MLP). Lastly, BiLSTM is preferred above ConvNet, seeing it outperforms in most used cases.	(Camacho-Collados & Pilevar, 2017; Han & Akbari, 2018; Poolsawad et al., 2014; Revina et al., 2020; Kowsari et al., 2019, 2017)

Chapter 4

Project Context

This chapter describes the nature of the dataset I work with, Section 4.1. Followed by an analysis of the problem context which is described in Section 4.2.

4.1 Dataset

The dataset consists of over 500.520 technical support tickets, classified into 114 different ticket categories. 71 categories contain less than 500 tickets and 7 others are too broad e.g., "Inquire / Help", "Request for information", see Table 4.1. Together, these account for 41% of the dataset. The ticket data has an additional caveat of being multilingual. These problems are handled in the preprocessing stage by applying a language detection model (Salcianu et al., 2020). During this process, three fields are checked if they are written in English, "Short Description", "Description", and "Resolution / Work Notes". Only the English written fields are combined. This form of text representation is chosen, instead of providing three inputs to a model or only one field. Combining the fields ensures that less tickets are discarded during the language detection (as shown in Table 4.2).

The process that a ticket goes through before reaching the models is as follows:

(1) Atos customer's employee (requestor) registers a ticket or contacts an Atos employee who registers the ticket;
(2) an Atos employee validates the ticket and completes the short description & description;
(3) the ticket is solved and work notes are added;
(4) a comment is added in which the solution is described, subsequently, it is validated by the requestor;
(5) the ticket is closed.

Furthermore, the written text, of both parties, is unstructured containing special characters, dates, typos, consecutive email-addresses, and line-breaks.

Table 4.1: Snapshot of the dataset

Account	Open / Closed Month	Category	SubCategory	Short Description	Description	Resolution / Work Notes
Account 1	May-21	Inquiry / Help	NaN	#SAFENET – Mobilepass (Resend Email) - User ne...	User's name: MAYUR PATIL_x000D_\nUser's contac...	2021-05-01 00:16:41 - Mylene Bajamundi (Work n...
Account 2	Apr - 2021	Request for Information	NaN	Request information on first login	Запрос информации по первому входу	Информация предоставлена в звонке.
Account 5	May-21	Access	Password/PIN	SD - Quantum Password Reset	User needs to get their domain account passwor...	Reset users password and user able to change a...
Account 5	May-21	Hardware	Faulty	FITS - EUCS - Hardware Fault - (No battery det...	For microphone/camera issues: see KB0014389\n\...	NaN
Account 5	May-21	Software	Configuration	Mapping a Shared Drive (FTF)	User ID:_x000D_\nDevice CA015079_x000D_\nDrive...	User was receiving an error message when tryin...
Account 6	Apr - 2021	PLANT_SAFETY	DOOR_ACCESS_CONTROL_SYSTEM	Entrance Tourniquet Malfunction	Raul Garcia\tEPJRX\t\tAvail: Business Hours\tM...	Lectoras de torniquete se encuentran funcionan...
Account 6	Mar - 2021	EDM_DI	DI_PLATFORM	macbook - Ip Config	CWID: GMCJY_x000D_\nPhone number: +49 173 79 4...	Incident set to resolved after the final remin...

Table 4.2: An overview of how many fields per ticket are written in English

number of fields	frequency
0	39.931
1	70.922
2	186.489
3	203.178

4.2 Problem Analysis

A classic data analysis involves the data analytic team to perform an exploratory data analysis of the categories to get an initial overview of the problem. Thereafter, a team is designated to classify the tickets into predetermined types. They establish the type based on the information the ticket provides, such as; (1) category, (2) subcategory, (3) short description, (4) description, and (5) resolution / work notes. The potential set of remediation types is dependent on the topic, which makes it beneficial to group tickets. However, grouping is only possible when the category and subcategory are accurate. Having the tickets clustered, directly improves the classification, since they have a better demarcated set to choose from, resulting in a less error prone environment. But there are some general problems relating to the classification by an agent:

(1) some fields are written in a language other than English, translation errors occur and it is harder to classify;
(2) tickets are discarded as irremediable, caused by skipping the ticket;
(3) the description could be too extensive or too short, resulting in an incorrect allocation.

These issues are considered errors since it could have been prevented by proper labeling of the tickets. Wrongly assigned remediation types, when large enough, have an effect in the analysis. In turn, this can lead to advising a different improvement to the customer, and letting the real issue continue.

Thus, artificial intelligence based methods for predicting categories is considered crucial for the accuracy of analysis in the long-term and increasing the EX. For only a fraction of the time a model needs to categorise tickets, the agent processes the tickets faster with these categories, thus clearing the agents time to focus on other tasks. Consequently, reducing the mistakes by the agent, which improves the output.

Summarising the problem would make it a document classification problem where the ticket's combined field is illustrated as a document and the ticket label as the document label. Seeing that the problem can be expressed as a document classification problem, it follows that the typical steps can be used for classifying technical support tickets. However, there are three types of text classification based on what the predicted result should be:

(1) multi-label classification, problem where multiple labels may be assigned to each instance;
(2) multi-class classification, problem of classifying instances into one or more classes;
(3) binary classification, the task of classifying the elements of a set into two groups, any prediction can contain either one of those classes.

The task is to assign one category to each ticket which would make it a multi-class classification task, where the technical support tickets are classified into 114 different ticket categories (e.g. Hardware, Software, Applications, Event management, etc.).

Chapter 5

Methodology

In this chapter I highlight the dataset and the preprocessing steps applied to end up with tokenized data, Section 5.1. Followed by Section 5.2, where the steps and models used in this study are described.

5.1 Dataset Preparation

Before a text classification model can be applied, the first step for building a model is data preprocessing. This step addresses the issue of vocabulary size and noise found in the input documents by either reducing the vocabulary size or removing noise. This is anticipated to help in maximising the classifier's performance (Krouska, Troussas, & Virvou, 2016; Barushka & Hájek, 2019).

Noise in natural language text can be one of the following; character repetitions, non-standard words, spelling errors, missing punctuation, abbreviations, etc.. In this study, I performed the commonly used techniques in text preprocessing, explained in Chapter 3, along with domain-specific operations that are based on the ticket descriptions and exploratory data analysis, such as label remapping (Section 5.1.1) and language filtering (Section 5.1.2), as explained in Chapter 4. Inspecting the structure of the ticket, Table 5.1, lets me reduce the size of the ticket by only maintaining the fields that hold valuable information. These are 'short description', 'description', 'resolution/work notes', and their corresponding category. The other fields are removed for the following reasons:

(1) account, the model is intended to predict based on the written text. Information regarding who it originates from is not desired, since the model could learn different word representations based on customers;
(2) open/closed month, when the ticket got closed hold no information regarding the topic of a ticket;
(3) subcategory, is more specific than category. However, it has the same problems that category experiences. Elements of this field were researched by domain experts to enrich the category field, yet, without any success.

An additional domain-specific step in preprocessing is generalising the category labels. The dataset is from six customers, each with their own categories, some more useful than others. Therefore, a domain expert and I applied category remapping to combine

Table 5.1: Technical support ticket data

Account	Open / Closed Month	Category	Subcategory	Short Description	Description	Resolution / Work Notes
Account 1	May-21	Inquiry / Help		Unable to con...	Unable to con...	2021-05-01 02:15:38...
Account 5	Jun-21	Software	Configuration	SD - Ras Token	**See KB001538...	User has no ..

similar topics, described in Section 5.1.1.

A summary of the preprocessing steps that are executed are below:

1. remapping labels and removing irrelevant ones;
2. language detection and filter out non-English fields;
3. remove characters with regular expressions; lowercase, numbers, line breaks, punctuation;
4. tokenization;
5. lemmatisation;
6. remove stop-words;
7. threshold word count;
8. discard tickets with missing data;
9. training and test set.

5.1.1 Label correction

Chapter 4 described the preliminary findings of the dataset, such as class imbalance, polylingual, and three written fields. Originally the dataset contained 114 classes. To end up with a model that would be optimised for relevant categories, some classes had to be excluded. The following steps are taken in consultation with two domain experts, one being the client:

(1) the first step was to remove classes that were ambiguous (e.g., "Inquiry / Help" and "Incident");
(2) thereafter, similar classes are combined, that were named differently by each customer (e.g., "applications", "APPLICATION");
(3) lastly, some labels were customer-specific ("INTERGRATION_MONSANTO") and seeing that the model is intended for categorising the majority of the tickets, those were removed.

As a result 12 labels are left, to be classified by the model, using the remapping illustrated in Table 5.2. This step reduces the number of tickets from 500.520 to 271.723.

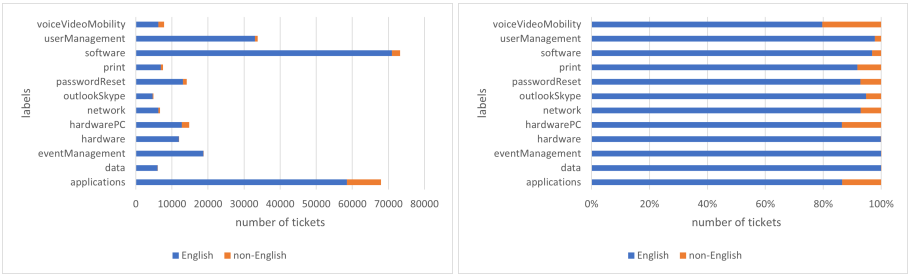
5.1.2 Natural language filter

The research is more focused on the applicability of text classification on technical support tickets. I therefore carefully examined the list of discarded fields during the step of removing non-English descriptions. To my disappointment a lot of language detection models (CLD3[1], TextBlob[2], Polyglot[3], langdetect[4], FastText[5]) classified short text, primarily containing nouns, as other than English. Figure 5.1 gives an overview of how 18.659 tickets are removed and were distributed over the labels. Not knowing the

[1]Compact Language Detection v3
[2]TextBlob: Simplified Text Processing
[3]Polyglot
[4]Nakatani Shuyo's language-detection
[5]FastText

Table 5.2: Overview on the new labels, old categories and their ticket count

New category	Original category	no. Tickets
applications	APPLICATIONS	39.462
	Application	12.742
	Applications	17.792
data	Data	6.018
eventManagement	EVENT_MANAGEMENT	18.765
hardware	Hardware	11.972
hardwarePC	Desktop / Laptop	7.640
	PC_HARDWARE	7.483
network	NETWORK	4.968
	Network	1.803
outlookSkype	OUTLOOK_SKYPE	5.033
passwordReset	PASSWORD_RESET	14.339
print	PRINT_FAX	3.460
	Print	4.232
software	SOFTWARE	22.021
	Software	48.629
	WORKPLACE_TOOLS	3.192
userManagement	Access	27.219
	USER_MANAGEMENT	6.625
	User Management	173
voiceVideoMobility	VOICE_VIDEO_MOBILITY	8.155



(a) number of tickets (b) % of total

Figure 5.1: Distribution natural language among labels, 5.1a showing the absolute difference and 5.1b the distribution per category

exact contribution of non-English descriptions makes it difficult to measure the impact. The label where it could have the largest impact is 'voiceVideoMobility', reducing the number of tickets by almost 26%. Further analysis on 50 randomly picked samples of that label reveals that the language detection did filter out non-English descriptions, making it even more difficult to assess the significance of removing wrong classified descriptions. Therefore, I decided not to put effort into filtering wrongly classified descriptions, since there are plenty of tickets left, 253.064 in total.

5.1.3 Remove characters

The next step in text preprocessing is to examine the text for peculiarities. After auditing the ticket description with 100 random samples, and with 50 smallest and 50 largest written tickets, nine findings are reported:

1 excel line breaks, "_x000d_", the data is loaded in using Excel files, which generates these line breaks to indicate it needs to be placed in a new row. They do not contain information on the subject of the ticket;
2 escape sequences, "\n\r\t", these three options represent new line, carriage return and tab, respectively. They have the same importance as Excel line breaks;
3 web address, the URLs are different for each customer. Although, it does say something about the problem or solution, it is something that changes overtime which a model cannot compute;
4 single and consecutive mail addresses, the description inside a mail is important, not who it was sent to or from. The model should be trained on what the problem is, not for who it is intended;
5 special characters, such as bullet points and long dashes, these are not filtered out during the removal of punctuation. Nevertheless, they are not significant whatsoever;
6 timestamp, mm/dd/yyyy and hh:mm:ss, when a ticket is picked up does not contribute to the description of the topic. Therefore, timestamps are filtered out;
7 text in brackets, <..>/[...]/{...}/(...), some of the things inside brackets are; date, identifier, additional note, and HTML code. Considering, most of the occurrences are unwanted and only a small part of additional notes are helpful, the decision is made to remove brackets from text;
8 identifiers of employees, AB0123456, the model could in theory establish connections between certain identifiers and categories. Often, a ticket ends up at a particular division within the company. This is undesirable, as the model needs to predict the category based on the problem description. A drawback is the removal of IT4You and other nouns in that format;
9 lowercasing, although it is not strange for text to have capital letters, transforming every letter to its lowercase component reduces the vocabulary size and helps the model, since it does not see 'the' and 'The' as identical.

A drawback of removing identifiers and numbers is that some domain-specific words were removed in the process. For example, Windows 10 and Windows 2018 are truncated to Windows, but, one is a desktop and the other a server environment, respectively. Instead of creating a list of words that should be kept, a decision was made against it. The list would definitely be incomplete considering the diversity of the categories and the sheer amount of tickets, which would make it unfair and biased.

Table 5.3: Dataset statistics

Class	Tickets		Words					
	frequency	% of total	mean	median	min	max	vocabulary	top
applications	58.512	23.5	113	62	6	4.686	84.020	email
data	6.014	2.4	109	100	6	712	11.695	detail
eventManagement	18.747	7.5	64	64	6	1.179	3.364	event
hardware	11.950	4.8	218	184	6	1.148	16.621	device
hardwarePC	12.743	5.1	121	51	6	3.986	25.247	ticket
network	6.204	2.5	133	61	6	4.507	18.691	ticket
outlookSkype	4.630	1.9	73	53	6	2.669	13.644	email
passwordReset	13.073	5.3	40	30	6	1.243	15.792	password
print	6.902	2.8	133	155	6	874	13.726	printer
software	70.943	28.5	107	86	6	3.354	59.292	detail
userManagement	33.047	13.3	86	81	6	1.158	22.883	password
voiceVideoMobility	6.205	2.5	51	36	6	987	15.013	phone

5.1.4 Tokenization, lemmatisation & stop-word removal

A bigger concern is the removal of stop-words. During this process, words such as *password*, *access*, *device*, *reset* and *error* were removed.

This could have two consequences. First, the Natural Language Toolkit's (Garette et al., 2021) corpus that is used consists of a delimited list of dictionary words, in other terms, the words are stored in their lemma form. Second, this corpus is not a comprehensive list of English words, therefore words might be absent. To resolve the first problem of singular form words, lemmatisation is applied which ensures that words are kept in their dictionary form, known as lemma. Because stop-words are checked on the base form of the words, the step of lemmatisation is done prior to removing stop-words. To combat the latter problem, a list is created containing the words that should be kept.

5.1.5 Word count & missing data

As the description field should clearly not be empty, these tickets are omitted. Removing those tickets leads to the minimum text length of one. However, looking at the description of tickets with equal or less than five words leads to two observations:

(1) the text has little to no information, e.g., "various issues";
(2) the description contains only placeholders, such as "closed by caller";
(3) if the ticket contains information, it can hardly be placed in a specific category, e.g., "kathleen noone login ad" labelled in "hardwarePC".

In the case of classifying tickets, these cannot be used for training. There is just too little information to allow a classification based on those descriptions. The final size of the dataset is 248.970 tickets, which is 49.7% of what was available and has the following most common words shown in Figure 5.2, and characteristics displayed in Table 5.3.

5.1.6 Training and test set

The split between the sets, training and test, is the same for all used cases. The training set contained 85% of the data and test set the remaining 15%. As pointed out in Chapter 4 the categories are imbalanced. To ensure that the classes are represented

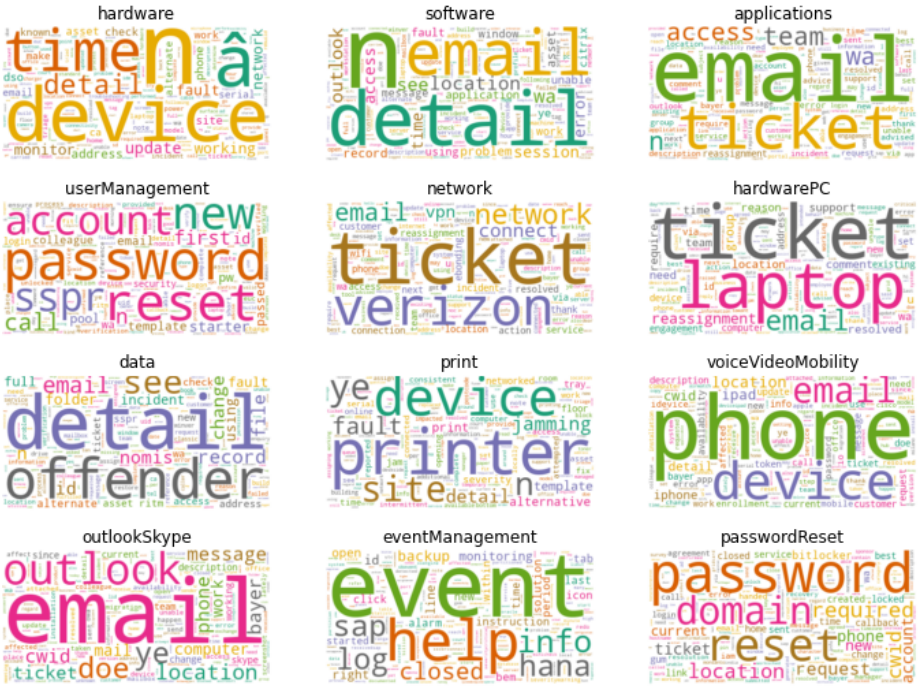


Figure 5.2: Word cloud displaying the most frequent words per class

evenly between the two sets, a stratified shuffle is used to split them. The test set is only used at the final validation of the models, while the training set can be divided into a validation set when necessary. The same ratio is applied in those cases, 85% training and 15% validation. However, these correspond to;

(1) training/validation/test split;

(1)(1) training, 72.25% or 179.880 tickets;
(1)(2) validation, 12.75% or 31.744 tickets;
(1)(3) test, 15% or 37.346 tickets;

(2) training/test split;

(2)(1) training, 85% or 211.624 tickets;
(2)(2) test, 15% or 37.346 tickets.

5.1.7 Outliers

Analysing Table 5.3, shows a few interesting characteristics:

(1) the class 'eventManagement' stands out in regard to vocabulary size, compared to the second smallest it is 3.5 times smaller and 25 times smaller compared to the class with the largest vocabulary; Considering the amount of tickets, it sits in the middle, this could derive from a good defined problem, making it simpler to model.
(2) the two largest classes ("applications" and "software"), have by far the largest vocabulary, at least twice as big compared to the next successor. This could indicate that the topic consists of a wide range of problems;

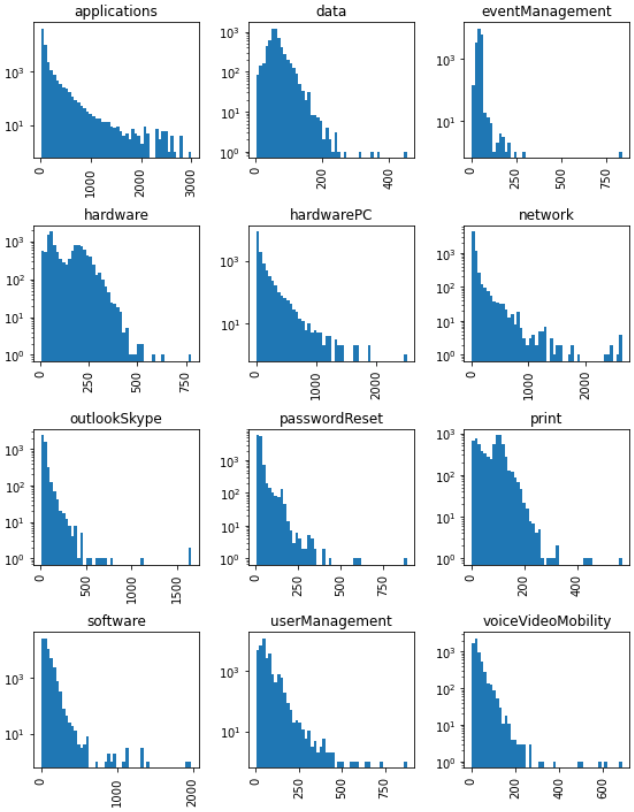


Figure 5.3: Twelve histograms, showing the distribution of the number of words per ticket. Y-axis is the frequency on the logarithmic scale and X-axis the number of words in a ticket

(3) some of the categories share their most common words, Figure 5.2, which could signal a weak definition of the topic or that they overlap on a few of the issues;
(4) most classes have less words per ticket than their average, except "print". This is reinforced in Figure 5.3, where it is one of the two classes with a second peak;
(5) lastly, all of the tickets are strongly represented up to the 150 word range, then declining. Every class has a few outliers regarding the number of words in a ticket.

5.2 Empirical Study

This section describes the empirical study. In particular, the feature engineering (Section 5.2.1), models and their used infrastructure (Section 5.2.2), and lastly, the validation measures (Section 5.2.3).

5.2.1 Feature engineering

Feature engineering is essential in our use case, since two different tactics are used, (1) TFIDF, and (2) word embedding.

Term frequency-inverse document frequency

TFIDF is a combination of two metrics, term frequency (tf) and inverse document frequency (idf). The former is a measure of how frequently a term appears in a document and the latter is a measure of how important a term is. It is computed by dividing the total number of documents in the corpus by the document frequency for each term and then applying logarithmic scaling to the result. The formula for calculating the TFIDF is,

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{5.1}$$

$$\text{idf}(t, D) = \log \frac{1 + N}{1 + |\{d \in D : t \in d\}|} + 1 \tag{5.2}$$

with:

- $f_{t,d}$, number of times a term t occurs in a document d;
- $\sum_{t' \in d} f_{t',d}$, number of words in document d;
- N, total number of documents in the corpus, $N = |D|$;
- $|\{d \in D : t \in d\}|$, number of documents where the term t appears. If the term is not in the corpus, this will lead to a division-by-zero. That is why scikit-learn uses idf smoothing variant, which adds a '1' to the numerator and denominator, which corresponds to having a document that contains all terms.[6]

For TFIDF, an existing function is used of scikit-learn, named TfidfVectorizer[7]. This function has a few parameters that can be tuned:

[6]scikit-learn TFIDF implementation
[7]scikit-learn feature extraction for text, TFIDF

(1) ngram_range, the lower and upper boundary of the range of n-values for different n-grams to be extracted. An n-gram is a sequence of n words and is a type of probabilistic for predicting the next item in such a sequence in the form of a $(N-1)$ order Markov model (Jurafsky & Martin, n.d.). However, the goal is not to predict the next word. Nevertheless, applying n-grams could help the model to establish connections between grouped words and categories, e.g., "microsoft outlook" and "outlookSkype";
(2) max_df, when building the vocabulary, it ignores terms that have a document frequency higher than the threshold;
(3) min_df, the opposite of max_df, when building the vocabulary, it ignores terms that have a document frequency lower than the threshold;
(4) max_features, the maximum number of features to keep, based on word frequency. Only the most common features will be kept.

This feature-space function is used in DNN and SVM, known as a document term matrix (DTM), and it disregards grammar and even word order but keeps multiplicity.

Word embedding

The other word representation is word embedding. Considering that in the literature, Chapter 3, pretrained word embeddings did not perform well on domain-specific tasks, word embedding is applied by adding an embedding layer in the neural network. Nonetheless, two steps need to be performed to transform the data into an input format the model can use; (1) tokenization and (2) sequence padding or truncating.

Text tokenization, the written text is tokenized, which is converting sentences to sequences where each word has a numeric representation, for example; "The quick brown fox jumps over the lazy dog" and "The quick brown fox was jumping over the lazy dog" are converted to [1, 2, 3, 4, 5, 6, 7, 8, 9] and [1, 2, 3, 4, 10, 5, 6, 7, 8, 9], respectively. This is mandatory, as the model cannot process text and no information is lost when each word has their own numeric representation.

The second step, sequence padding, is to pad or truncate each sequence to a fixed length. Padding comes from the need to encode sequence data into contiguous batches, to make all sequences in a batch fit a given standard length. These two steps have a few parameters that can be tuned[8][9]:

(1) num_words, the maximum number of words to keep, based on word frequency. Only the most common words will be kept;
(2) maxlen, maximum length of all sequences;
(3) truncating, removes values from sequences larger than 'maxlen', either at the beginning or at the end of the sequences.

Padding is a variable that can be set to either pad before or after each sequence. However, Dwarampudi and Reddy (2019) state that for LSTMs, pre-padding shows better accuracy. Therefore, the variable is left out in the list, since the impact will not be studied.

[8]The function Tokenizer is used to transform the text.
[9]The function pad_sequences from tensorflow is used to truncate or pad the sequence.

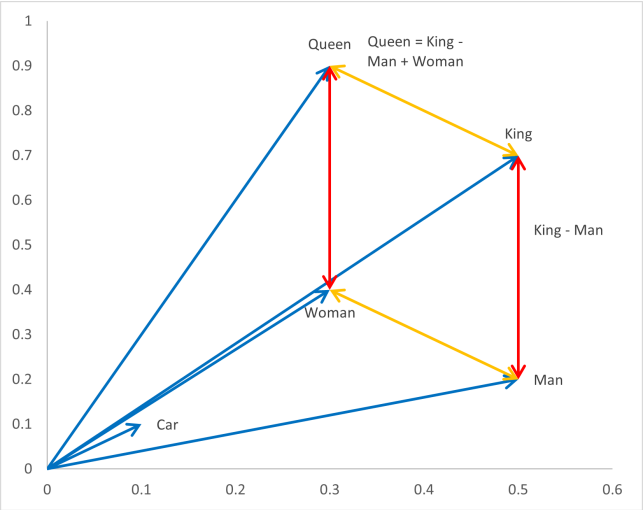


Figure 5.4: vector representation of a word embedding consisting of; king, queen, man, woman and car

Now, text is transformed to sequences that can be used by the model. However, sequences are not processed, but word embeddings. To create a word embedding, an embedding layer is added in the neural network, which feeds the BiLSTM word embeddings.

On the side of the neural network, the parameters related to the embedding layer are; (1) size of the embedding (embedding_size), (2) size of the vocabulary (max_words), and (3) length of the sequence (input_length). As it is not the intention to restrict the word embedding, the variable 'max_words' is set to an arbitrarily high number. Word embeddings are used in RNN and DNN. DNN will be used twice, since it is also used in combination with TFIDF. This decision is made, in order to compare the impact of using word embeddings, which in theory should maintain the structure of the sentence and capture the semantics of words.

An embedding layer transforms each numeric value to a vector representation of n dimensions. The idea is that words with similar definitions are put closer in vector space, i.e., 'king' and 'queen' are closer to one another than 'car' and 'king', a common analogy is King $-$ Man $+$ Woman $=$ Queen. To make it more explainable, consider an embedding of 2 dimensions, (x, y). Figure 5.4 shows that the distance between related words are the same. The similar direction of the red and orange arrows indicates similar relational meaning. In our case, words are most likely grouped based on the category. Because we are not using methods as skip-gram, continuous bag-of-words model, singular value decomposition (hereinafter SVD), or positive pointwise mutual information (Lai et al., 2016) to learn semantics, morphological, context, or hierarchical information.

Most of the aforementioned methods operate on the basis of matrix factorisation (Levy & Goldberg, 2014). Matrix factorisation is a way to generate latent features when multiplying different kinds of entities. As an example, the problem is described using SVD,

using the matrices in Table 5.4 (Liang, Altosaar, Charlin, & Blei, 2016; Mikolov, Chen, Corrado, & Dean, 2013). The SVD of a matrix is a factorisation of that matrix into three matrices. It conveys geometrical and theoretical insights about linear transformations.

Table 5.4: The four matrices for the example in singular value decomposition

C	d_1	d_2	d_3	d_4	d_5	d_6
settings	1	0	1	0	0	0
password	0	1	0	0	0	0
options	1	1	0	0	0	0
memory	1	0	0	1	1	0
disk	0	0	0	1	0	1

V	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

U	1	2	3	4	5
settings	-0.44	-0.30	0.57	0.58	0.25
password	-0.13	-0.33	-0.59	0.00	0.73
options	-0.48	-0.51	-0.37	0.00	-0.61
memory	-0.70	0.35	0.15	-0.58	0.16
disk	-0.26	0.65	-0.41	0.58	-0.09

W^T	d_1	d_2	d_3	d_4	d_5	d_6
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

Where:

(1) C: being the co-occurrence matrix. One row per word, one column per min(M, N) where M is the number of words and N is the number of documents;
(2) U: think of the dimensions as "semantic" dimensions that capture distinct topics like politcs, sport, economics. For example, dimension 2 being hardware/software. Each number in u_{ij} in the matrix indicates how strongly related word i is to the topic represented by semantic dimension j;
(3) V: this is a square, diagonal matrix of dimensionality min(M, N) $\times$ min(M, N). The magnitude of the singular value measures the importance of the corresponding semantic dimension;
(4) W^T: one column per document, one row per min(M, N). These are the semantic dimensions from matrices U and V that capture distinct topics. Each number w_{ij} in the matrix indicates how strongly related document i is to the topic represented by semantic dimension j;

SVD is the decomposition of C into a representation of the words (U), a representation of the documents (W^T) and a representation of the importance of the semantic dimensions (V).

$$C = UVW^T \tag{5.3}$$

Having these representations makes it possible to perform dimensionality reductions and similarity checks between words, semantic dimensions or documents, e.g., reducing the dimensionality to two results in the following set of matrices, showed in Table 5.5.

Table 5.5: The four matrices of SVD with a dimensionality reduction

C_2	d_1	d_2	d_3	d_4	d_5	d_6
settings	0.85	0.52	0.28	0.13	0.21	-0.08
password	0.36	0.36	0.16	-0.20	-0.02	-0.18
options	1.01	0.72	0.36	-0.04	0.16	-0.21
memory	0.97	0.12	0.20	1.03	0.62	0.41
disk	0.12	-0.39	-0.08	0.90	0.41	0.49

V_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

U	1	2	3	4	5
settings	-0.44	-0.30	0.57	0.58	0.25
password	-0.13	-0.33	-0.59	0.00	0.73
options	-0.48	-0.51	-0.37	0.00	-0.61
memory	-0.70	0.35	0.15	-0.58	0.16
disk	-0.26	0.65	-0.41	0.58	-0.09

W^T	d_1	d_2	d_3	d_4	d_5	d_6
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

When computing the similarity between d_2 and d_3 for the original matrix (C) and for the reduced matrix (C_2), a difference is noted, because the dimensionality is reduced:

- similarity of d_2 and d_3 in the original space is 0;
- similarity of d_2 and d_3 in the reduced space is $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$.

SVD is a matrix-based algorithm for word embedding, originating from latent semantic analysis (Levy, Goldberg, & Dagan, 2015).

5.2.2 Architecture

The models used in this study are described below:

1. support vector machines (SVM) (Noble, 2006), this is a supervised machine learning model based on the statistical learning framework proposed by Vapnik and Chervonekis (Cherkassky & Mulier, 1999). It is a robust model often used for text classification and less affected by the class imbalance problem (Yanminsun et al., 2011; Joachims, 1998; Telnoni, Budiawan, & Qana'a, 2019). SVM is used as a baseline for what can be achieved with a more complex model. The linear algorithm of support vector classifier (SVC) was chosen, the reason is that the computation time is much lower than the other kernels. Since the other kernels are forced to apply a 'one-vs-one' strategy in case of a multi-class strategy, instead of the 'one-vs-rest' of a linear strategy. LinearSVC[10] of the scikit-learn package is used, this function has a few parameters that can be optimised:
    - penalty, specifies the norm used in the penalisation;
    - loss, specifies the loss function;
    - class_weight, adjusts the weights according to class frequencies.
2. deep neural network (DNN) (Kowsari et al., 2017), this is a MLP which is designed to learn through multi-connection of layers where every single layer only receives the connection from the previous layer and provides connections only to the next layer in a hidden part, illustrated in Figure 5.5 (Kowsari et al., 2019). Although a neural network is more computation heavy, it is flexible with the preprocessing of the features which make it more robust (Revina et al., 2020). The structure of MLP changes when word embedding is required. An embedding layer is added at the beginning of the model, so a domain-specific word embedding can be trained. Two variances of the model are constructed, an example of both is shown in Figure 5.6. The design of the model is somewhat flexible, as well as the number of hidden layers and how many nodes each layer has. In total five hyperparameters are tested (Kowsari et al., 2019):
    - batch_size, number of samples per gradient update;
    - hidden_layers, number of dense layers within the model;
    - dropout, fraction of the input units to drop;
    - dense_nparams, dimensionality of the output space;
    - optimiser, an algorithm that modifies the attributes, such as weights and learning rate.

[10] svm, LinearSVC

Hyperparameters that are not examined:

- loss function (categorical cross-entropy), a function used to evaluate a candidate solution, almost always one with the lowest score. Categorical cross-entropy is a multi-class loss function, it needs a softmax activation function on the last layer of the MLP, as long as the outputs are mutually exclusive, otherwise sigmoid could be used instead;
- activation function (rectified linear unit), defines the output of a node using a function. Another great function would be sigmoid, tanh or softmax. Nevertheless, rectified linear unit is picked, since it is quick to calculate and have better gradient propagation (Mhaskar & Micchelli, 1994).

3. recurrent neural network (RNN) (Sutskever, Martens, & Hinton, 2011), this architecture assigns more weight to the previous data points of a sequence. Therefore, this technique is a powerful method for text. A RNN considers the information of previous nodes in a very sophisticated method which allows for better semantic analysis. This study uses long short-term memory (hereinafter LSTM) (Hochreiter & Schmidhuber, 1997) for text classification. LSTM is a special type of RNN that preserves long term dependency in a more effective way, solving the vanishing gradient and exploding gradient problem that RNN typically has (Kowsari et al., 2019). LSTM trains left-to-right, but it is possible to process the data from right-to-left which makes it a Bi-directional LSTM (BiLSTM). This additional training capability is beneficial in providing better predictions (Siami-Namini, Tavakoli, & Namin, 2019). The input of a RNN is a sequence. An embedding layer is added at the beginning of the model, to train a domain-specific word embedding which is preferred, instead of using a pretrained word embedding (Wahba et al., 2020). BiLSTM could nullify the argument of pre- or post-padding, seeing that BiLSTM processes sequences in both directions. The structure of the BiLSTM is as pictured in Figure 5.7. After the BiLSTM layer global max pooling is implemented, Zemp (2021) states that global max pooling has a positive effect on the results, compared to global average pooling. Lastly, the hyperparameters of BiLSTM are:

   - spatial dropout, a type of dropout for embedding;
   - layers, dimensionality of the output space;
   - optimiser, an algorithm that modifies the attributes, such as weight and learning rate;
   - batch_size, number of samples per gradient update.

   However, these are not all the hyperparameters, TensorFlow has an implementation to quicken the training by almost twenty times. It does require fixed parameters, listed below:

   - activation, tanh;
   - recurrent_activation, sigmoid;
   - recurrent_dropout, 0;
   - unroll, false;
   - use_bias, true.

   Lastly, the loss function is not tested, categorical cross-entropy is used to evaluate a candidate solution. Categorical cross-entropy is a multi-class loss function, it

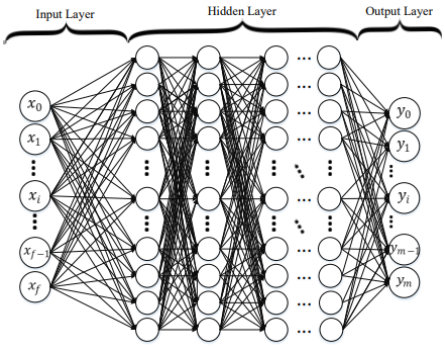
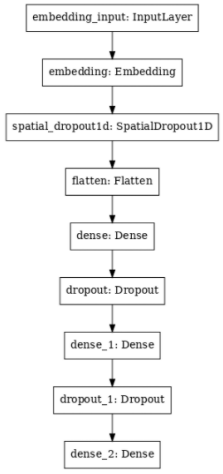
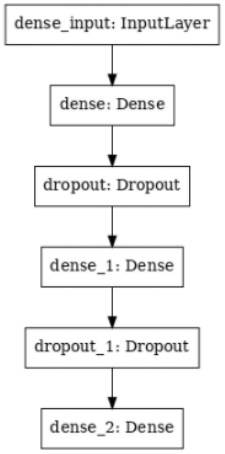


Figure 5.5: Standard fully connected deep neural network (Kowsari et al., 2019)



(a) Embedding + MLP, word embedding and 2 hidden layers



(b) MLP, 2 hidden layers

Figure 5.6: Architecture of a DNN

needs a softmax activation function on the last dense layer, as long as the outputs are mutually exclusive, otherwise sigmoid could be used instead.

5.2.3 Validation

To validate the performance of the previously mentioned models, the data classification measures (Hossin & M.N, 2015), Precision, Recall, Accuracy, Support and two F_1 scores are used. The F_1 score is the harmonic mean of precision and recall. As mentioned earlier in Section 4.2, the task encompasses a multi-classification problem, and suffers from class imbalance. Therefore, the weighted F_1 score metric is best suited since it takes support into account, which is the number of samples in a class. The macro F_1 score metric, which does not incorporate class size, only how many classes there are, is reported as well, as it provides information on the performance of the model regarding the average F_1 score. However, the weighted F_1 score determines how effective a model is in predicting the category of a ticket. The accuracy is provided to give information on

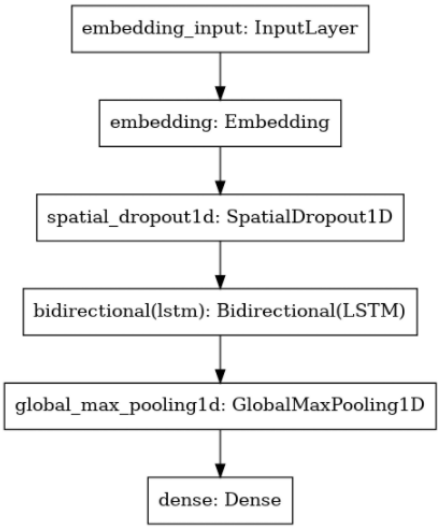


Figure 5.7: Architecture of a BiLSTM with an embedding layer

		Predicted			
	Classes	a	b	c	d
Actual	a	TN	FP	TN	TN
	b	FN	TP	FN	FN
	c	TN	FP	TN	TN
	d	TN	FP	TN	TN

Figure 5.8: Confusion matrix of a multi-class classification problem on the perspective of class b

how many tickets are correctly predicted. Accuracy does not consider the performance per class. For multi-class classification, Accuracy can be calculated using the micro F_1 score, which corresponds to micro-Recall and micro-Precision (Grandini, Bagli, & Visani, 2020). Both Precision and Recall are calculated using a confusion matrix. Support is the number of occurrences of each class. A confusion matrix is a summary of prediction results, for a multi-class classification problem it would have the following portrayal as shown in Figure 5.8. The metrics are as followed computed:

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}} \tag{5.4}$$

$$\text{Recall} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FP)}} \tag{5.5}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.6}$$

$$\text{F}_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.7}$$

[10]Micro looks at the confusion matrixs as a whole (not per class), resulting in equal values.

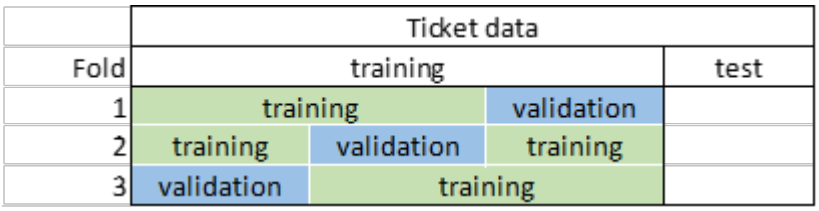


Figure 5.9: 3-fold cross validation

$$\text{weighted } F_1, \text{macro } F_1 = \frac{\sum F_1 * \text{Support}}{\sum \text{Support}}, \frac{\sum F_1}{\#\text{Classes}} \tag{5.8}$$

The validation of each model with the aforementioned metrics undergoes two stages; (1) tuning of the hyperparameters, and (2) final validation of the model.

For tuning the hyperparameters, cross-validation is used which consists of averaging the prediction estimates of K train-test splits, where each data point is only used in a single test set (Moss, Leslie, & Rayson, 2018; Kohavi, 1995), shown in Figure 5.9. The model is trained on the training set and scored on the validation set. The process is repeated until each unique group has been used as the validation set. Stratified cross-validation is used, which preserves the percentage of samples for each class. Three splits are created that resembles about 67% of the earlier split is used for training and 33% is used for validation. Where possible, the cross-validation is done through grid search. Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. In particular, GridSearchCV[11] is used which has a built-in cross-validation. Aside from averaging the prediction estimates, the standard deviation of the prediction estimates provides crucial information on the stability of the model. Each model has their own set of hyperparameters, tuning all of them would take forever. Thus, the chosen hyperparameters have the biggest impact in the models' performance, resulting in the following list:

- support vector machines (SVM);
  - ngram_range, ((1, 1) or (1, 2)), unigram or a combination of unigram and bigram. As ngram is not used how it was originally intended, only unigrams and bigrams are validated;
  - min_df, (1, 2, 10), these have a significant impact on the vocabulary size, ranging from 169.469 with 1 applied, to 16.414 using 10. The aim is to conserve as much information as possible and be able to run the model. However, we have limited GPU memory which makes it impossible to use the entire vocabulary in combination with an n-gram range of (1, 2), creating a vocabulary of 1.866.145 words;
  - penalty, ('l1' or 'l2'), these are all the options that LinearSVC provides;
  - class_weight, ('None' or 'balanced'), the same constraint as with 'penalty', 'None' indicates that the classes have the same weight, whereas, 'balanced' calculates the class weights using the input data;
  - loss, ('hinge', 'squared_hinge'), both are 'hinge' loss functions. Except, the latter squares the 'hinge' loss, as the name suggests.

[11] scikit-learn GridSearchCV

- deep neural network (DNN):
  - using TFIDF:
    - batch_size (16 or 256), a higher batch size can be beneficial considering class imbalance. However, this needs to be tested so a relatively low and high batch size is selected. A value lower of 16 is not desired since the time to train drastically increases;
    - hidden_layers (2, 4 or 8), the articles that are found did not report how many hidden layers they used, therefore a random set is chosen;
    - dropout (0.1 or 0.3), Zemp (2021) reported to use a dropout of 0.1 in other models, so an additional value is tested to see if that is also the case here;
    - dense_nparams (64, 512 or 2048), the articles that are found did not report the amount of nodes they used per layer, therefore a wide range of values are tested;
    - optimiser (stochastic gradient descent or Adam), most articles have success using Adam as their optimiser. Instead of tuning the parameters of the optimiser, Adam will be compared with stochastic gradient descent.
  - using word embedding (Zemp, 2021);
    - truncating, ('pre' or 'post'), the effect could be negligible if the sequence length is equal to the max length that occurs in the corpus. However, this is not the case, truncating the first or last part of a text could remove information on the issue. The result will indicate where the information is written in the description;
    - maxlen, (64, 128 or 256), Zemp (2021) report that increasing the number of words above 300 had no effect. However, truncating it below 50 had a negative impact. Thus, values between these boundaries are tested;
    - embedding_size (64 or 256), Zemp (2021) report that the size of the embedding layer is key, as it improves the results significantly for smaller classes. Increasing the embedding layer above 200 had no effect, yet, an embedding size of 400 is used. Thus, a size above 200 and a smaller size is checked;
    - batch_size (16 or 256), see DNN using TFIDF.
- recurrent neural network (RNN) (Zemp, 2021):
  - truncating ('pre' or 'post'), see DNN using word embedding;
  - maxlen (64, 128 or 256), see DNN using word embedding;
  - embedding_size (64 or 256), see DNN using word embedding;
  - layers (64 or 256), Camacho-Collados and Pilevar (2017) used a size of 300, identical to Zemp (2021), but they do not substantiate that number. It seems quite high, so it is compared against a lower value;
  - spatial_dropout (0.1 or 0.3), see DNN using TFIDF;
  - optimiser (stochastic gradient descent or Adam), see DNN using TFIDF;
  - batch_size (16 or 256), see DNN using TFIDF.

The final validation is performed using the training and test set, this is the first time that the test set, or unseen data, is used. Since over-fitting is an issue in neural networks, an early stopper on the validation loss is implemented to avoid this phenomenon (Hawkins, 2004). As a result, the DNN and RNN training set is split into an extra validation set.

Chapter 6

Results

Chapter 5 described the methods of validation and the models. This chapter presents the results of the cross-validation and validation on the test set. Considering, that the vocabulary size fluctuates heavily per category, an overview of that effect is given, so it can be analysed in the discussion. As well as, the effect class imbalance has on the effectiveness of classifying the category.

6.1 Cross-validation

The top results obtained after experimenting with different hyperparameters are presented in the following Figure 6.1. The hyperparameters that resulted in the highest weighted F_1 score are displayed in Table 6.1, and the corresponding architecture in Appendix B.

(a) models using TFIDF

SVM	
Hyperparameter	Setting
ngram_range	(1, 2)
min_df	1
penalty	l1
class_weight	balanced
loss	squared hinge

DNN[a]	
Hyperparameter	Setting
batch_size	128[b]
hidden_layers	2
dropout	0.1
dens_nparams	512
optimiser	adam

[a] The TFIDF parameters of SVM are also used in the feature engineering of DNN.
[b] the original setting of 16 or 256, resulted in an 'out of memory' error. even with a batch_size of 1. Setting the min_df to 2 solved this issue. The options (16, 128) were tested

(b) models using word embedding

DNN[a]	
Hyperparameter	Setting
truncating	pre
maxlen	256
embedding_size	256
batch_size	256

RNN	
Hyperparameter	Setting
truncating	pre
maxlen	256
embedding_size	256
layers	256
spatial_dropout	0.3
optimiser	adam
batch_size	256

[a] The remaining hyperparameters, such as dropout, hidden_layers, etc. where taken from DNN using TFIDF.

Table 6.1: Hyperparameters settings that resulted in the highest weighted F_1 score

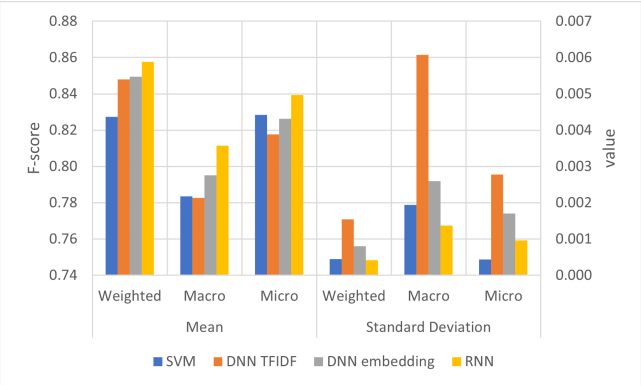
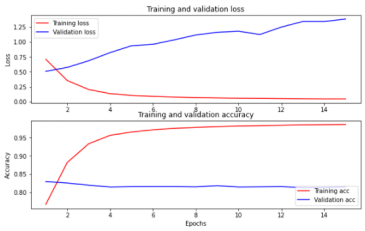
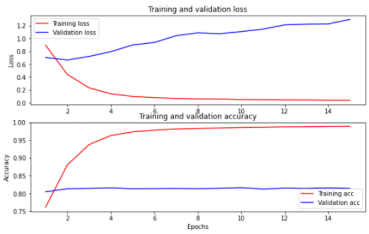


Figure 6.1: Cross-validation results of the best set of parameters

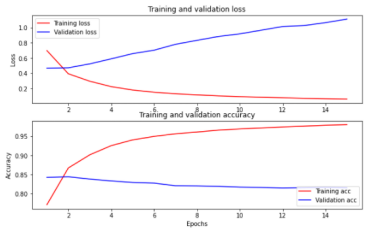
Figure 6.1 shows the different varieties of the F_1 score of each model and their corresponding standard deviations, calculated from the cross-validation results. Remarkably, the word embedding DNN and RNN, achieved a competitive weighted F_1 score of 85% and 86%, respectively, both outperforming the models using TFIDF. While all models achieved a low standard deviation, which is a stable performance across the cross-validation, SVM is the most consistent one, closely followed by RNN. SVM has a



(a) DNN + embedding



(b) DNN



(c) RNN + embedding

Figure 6.2: Neural Network training accuracy and loss development over 15 epochs

slightly higher accuracy (micro F_1 score) than the weighted counterpart, which means a lower bias towards majority classes.

6.2 Validation on the test set

After the cross-validation, each model is trained using the entire training set and their effectiveness rated on the test set, or unseen data. All of the neural network based models scored worse, DNN using TFIDF taking the biggest performance hit of 3.0% followed by DNN using word embedding by 1.8% and RNN with 1.1%. SVM had a small increase of 0.8%, which is expected, because the size of the dataset increased from 141.083 to 211.624 tickets. All models scored above 80% which was communicated by the client to be achievable. Table 6.2 shows the final F_1 scores of the four models. RNN reached the best results in terms of validation loss and validation accuracy in the second epoch, shown in Figure 6.2. After which, the model over-fits on the training set.

Table 6.2: F_1 scores of the four ticket classification models on the test set

	TFIDF		Word embedding	
Metric	SVM	DNN	DNN	RNN
Weighted F_1 score	0.8359	0.8175	0.8311	0.8463
Macro F_1 score	0.8021	0.7902	0.7939	0.8188
Accuracy	0.8362	0.8175	0.8319	0.8459

Since the dataset is imbalanced, there was an expectation that the classification algorithm would be biased favouring the major classes, leading to a high classification accuracy for the three major ticket categories; (i) software, (ii) applications, and (iii) userManagement, while showing poor accuracy towards the minor classes. Although

Weighted f1-score for RNN: 0.8462648478498562

	precision	recall	f1-score	support
applications	0.84	0.84	0.84	8777
data	0.94	0.85	0.89	902
eventManagement	1.00	0.98	0.99	2812
hardware	0.88	0.90	0.89	1793
hardwarePC	0.70	0.63	0.66	1911
network	0.72	0.69	0.70	931
outlookSkype	0.68	0.66	0.67	694
passwordReset	0.80	0.87	0.84	1961
print	0.89	0.88	0.89	1035
software	0.82	0.87	0.84	10642
userManagement	0.95	0.87	0.91	4957
voiceVideoMobility	0.68	0.70	0.69	931
accuracy			0.85	37346
macro avg	0.83	0.81	0.82	37346
weighted avg	0.85	0.85	0.85	37346

Figure 6.3: RNN Classification report of Word Embedding+BiLSTM showing precision, recall, f_1-score and support for all 12 classes

differences exist between the major and minor ones, shown in Figure 6.3 and Appendix A, it does not show the complete situation.

6.3 Category demarcation

The vocabulary size difference between categories prompted an examination of the relationship between vocabulary size, class size and performance, illustrated in Figure 6.4. The best predicted class is 'eventManagement' which was brought up earlier for its smaller sized vocabulary. Figure 6.4 shows no particular dependency between the three distributions.

However, properly defined categories, can be an argument for a smaller sized vocabulary. Analysing the confusion matrix of the predicted and actual class shows if categories potentially overlap, Table 6.3

Table 6.3: Confusion matrix of RNN predictions on the test set

Classes		Predicted											
		1	2	3	4	5	6	7	8	9	10	11	12
Actual	1	7634	1	1	2	268	61	32	91	18	576	29	64
	2	8	760	0	16	0	1	0	0	1	106	10	0
	3	31	0	2758	0	1	2	0	12	0	5	2	1
	4	4	13	0	1664	11	1	0	0	4	95	1	0
	5	396	0	0	6	1223	35	6	28	27	179	3	8
	6	196	0	0	2	48	588	1	8	4	69	5	10
	7	59	0	0	0	4	0	458	1	0	163	3	6
	8	108	0	1	0	12	6	1	1734	0	61	31	7
	9	49	0	0	3	7	3	0	0	914	55	1	3
	10	726	29	1	241	141	54	166	94	42	8979	34	135
	11	167	16	2	28	13	10	14	290	3	280	4132	2
	12	146	0	0	3	12	8	14	4	1	126	0	619

Legend	
1	application
2	data
3	eventManagement
4	hardware
5	hardwarePC
6	network
7	outlookSkype
8	passwordReset
9	print
10	software
11	userManagement
12	voiceVideoMobility

In the confusion matrix of RNN, it is noticeable that classes 'applications' and 'software' are large contributors in the misclassification of minor classes, it is even clearer

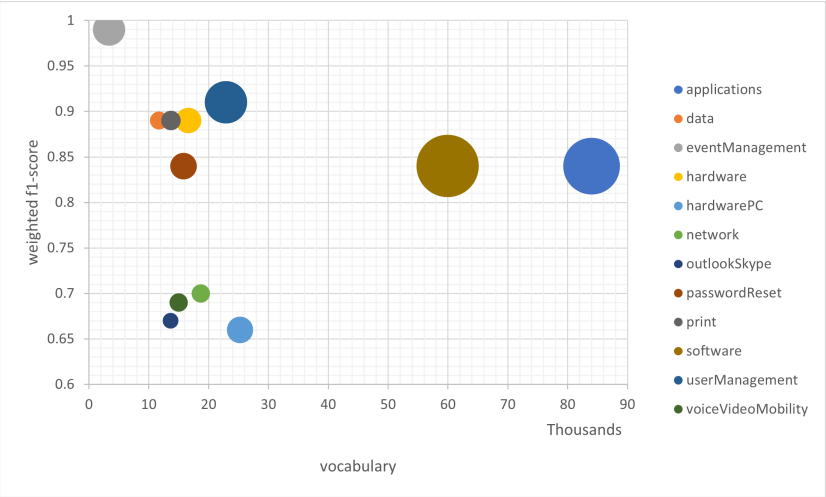


Figure 6.4: Plot between weighted f_1-score and vocabulary size per class with the size of the bubble representing the class size

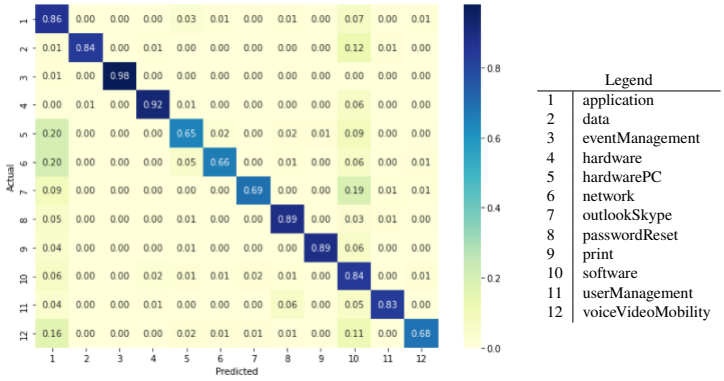


Figure 6.5: Normalised confusion matrix of RNN

illustrated in Figure 6.5. Either the problem description of the four worst scoring classes overlap with that of 'application' and 'software', or the class imbalance affect the classification of minor classes and predicts the majority class, when in doubt.

6.4 Class imbalance

As mentioned, the classes are not evenly distributed. The performance of the four models to classify each class is shown in Figure 6.6 It is clear that four categories ('voiceVideoMobility', 'outlookSkype', 'network' and 'hardwarePC') compared to the rest are not well predicted. Three of those categories have less than seven thousand tickets, making them the smallest classes. However, it is not clear if that is solely the reason for classification errors.

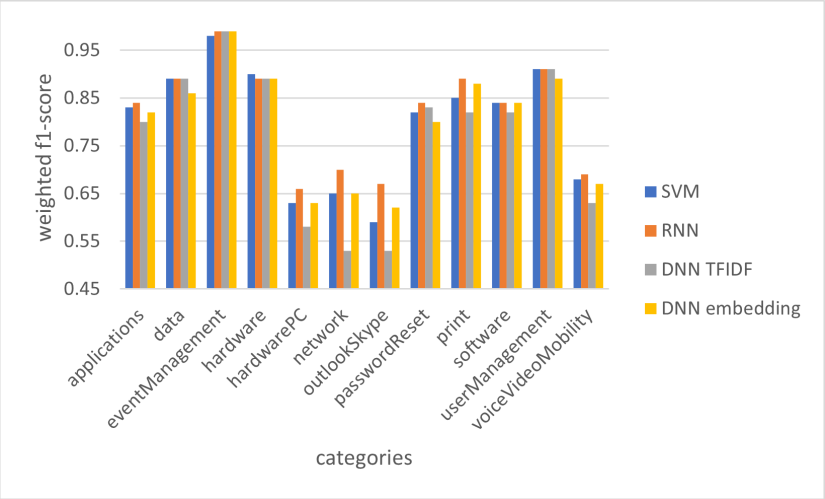


Figure 6.6: Classification score of the different models for all 12 classes

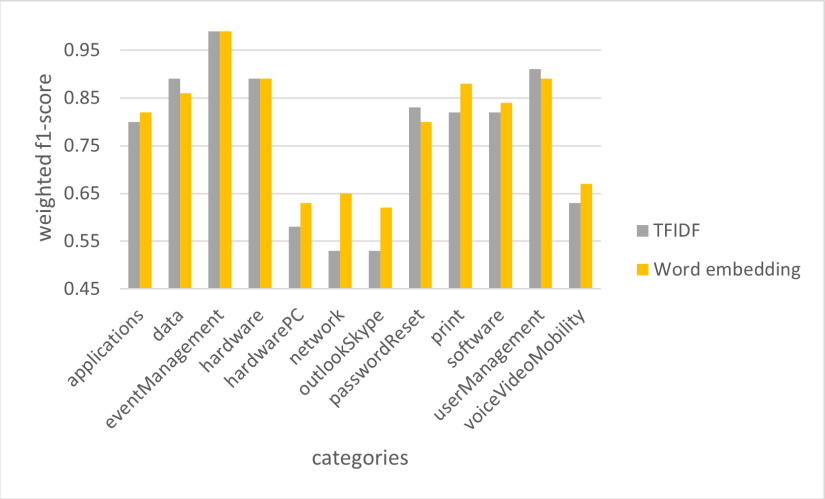


Figure 6.7: Classification score of DNN with different word representations for all 12 classes

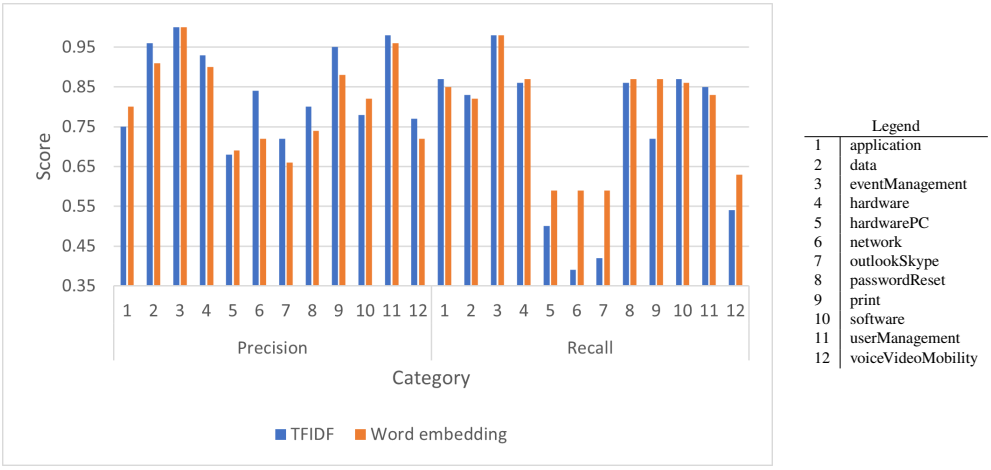


Figure 6.8: Precision and recall score of DNN with different word representations for all 12 classes

6.5 Word representation

Figure 6.7 shows a clear difference between the word representations when the weighted F_1 score is relatively low. Analysing Figure 6.8 shows that TFIDF has better precision scores, but word embedding exceeds TFIDF in the recall measure. Comparing the macro F_1 score makes it apparent, TFIDF scores 3% better in the metric precision but drops 6% in the metric recall, as shown in Figure A.3 and A.2. The performance of the models with word embedding to classify the four minor classes outperformed the models using TFIDF. The difference is smaller in classifying the major classes, but still noticeable and RNN is surpassing the other models as well, except for the category 'hardware' where SVM yields a better weighted F_1 score than RNN.

Chapter 7

Discussion

This chapter interprets the results and discusses certain aspects of this research, such as (1) the influence of preprocessing, (2) keeping the sentence structure by not merging the fields, (3) the results of feature engineering.

7.1 Text preprocessing

Starting with the first topic, in the cited literature text preprocessing seemed to have minor influence in the effectiveness of a model, except for domain-specific tasks. An assumption is made that classifying technical support tickets falls under that definition. Therefore this report incorporated certain text preprocessing steps that could have one of three effects, (1) positive, (2) pointless, or (3) negative. This research leaves that question open. However, it is important for Atos to know which steps to incorporate since the duration of the script determines how much the operation costs will be.

The results show two particularities that have two possible origins relating to preprocessing, where the particularities are lower F_1 scores in the minor classes and an almost perfect score for 'eventManagement'. The first suggests that the model would be bias towards major classes. The bias results in the model predicting the major class when in doubt. In the measurements this would be expressed as a high recall score but low precision score, since the model is predicting the major class even if it is actually the minor class. However, Figure 6.3 shows that the recall score of 'applications' and 'software' are lower than five other categories, suggesting that the model struggles in dividing certain problems among categories. This is also reflected in Figure 6.5, the prediction error is higher in only a few classes, indicating that those categories could overlap. Nonetheless, it cannot be ruled out that class imbalance did not affect the model in classifying categories. That 'eventManagement' almost had a perfect score in both word representations demonstrates that a proper defined category, results in a better predictor.

Class imbalance and overlapping categories are problems that are related to preprocessing. First, a good demarcated category increases the effectiveness considerably, so investing time into precise merging of customer-specific labels is advantageous. Lastly, class imbalance could have impacted some classes. Thus, as long as the training set is not large enough, future research should be done on data-space resampling techniques and their effect. The topic of data balancing is touched upon in the report, under the

assumption that the training set is large enough that class imbalance does not effect the performance. Therefore, data resampling yielded no benefit except potentially losing information.

7.2 Sentence structure

The ticket data consists of three written fields, these fields are combined to keep as much tickets as possible. However, this was on the premise that polylingual data would have a negative effect in the classification. In the scope of the research, it is stated that only English tickets will be processed. This forced the combining of these fields, to preserve as much tickets as possible. However, the impact is not measured and Zemp (2021) saw no effect in excluding other natural languages than English. Not removing non-English fields opens up the possibility to provide the model with three inputs, as a result, keeping the sentence structure of the fields, which could lead to a more effective prediction model.

7.3 Word representations

An important step in feature engineering is choosing the word representations, two are used in the research; (1) weighted word representation, term frequency-inverse document frequency, and (2) embedding layer. Zemp (2021); Wahba et al. (2020) studied tech classification on technical support tickets and used term frequency-inverse document frequency. When they apply this method in a comparable study, it withholds the use of categorical word representation i.e., one hot encoding and bag-of-words. However, it is noted that term frequency-inverse document frequency is not that meaningful with short written text. A hypothesis after analysing the characteristics of ticket data, was that the tickets contained enough words to justify the use of term frequency-inverse document frequency. Regarding the word embedding, a large corpus is often used to capture semantics. The neural network epoch graphs show why that is, as shown in Figure 6.2. The model adjusts the weight based on the training set, which looks normal. However, the metrics of the validation set, after the first or second epoch, is slowly getting worse, with an accuracy decrease of $\sim$1%. That the model reaches a near state-of-the-art result on the training set, an accuracy increase of +20%, indicating that the model tunes ticket specific words to be able to classify tickets in a category. Nevertheless, each used model was able to classify tickets based on the written data.

Chapter 8

Conclusion

The goal of this research is to find out if categories of technical support tickets can be accurately predicted. The research question is formulated to answer that objective and reads:

> *'How effective is a text classification model in classifying the category of technical support tickets, based on descriptive data provided by tickets?'*

The experiments show that classifying categories of technical support tickets are feasible. The written text provided by the ticket contains enough information for a model to predict a category. The results show that the effectiveness of text classification is affected by either, class imbalance or, overlapping categories. Six of the twelve categories are affected by this. These categories are; (1) applications, (2) hardwarePC, (3) network, (4) outlookSkype, (5) software, and (6) voiceVideoMobility. The category 'eventManagement' showed that a good demarcation of the topic led to a high weighted F_1 score. Word embeddings outperforms the document term matrix. The deep neural network with word embedding has a 1.4% higher weighted F_1 score than the one using the document term matrix. Recurrent neural network is the highest scoring model with a weighted F_1 score of 85%. It achieves this, by classifying the minority classes better. The recurrent neural network model is better in predicting the category than vector machine in all categories except 'hardware'. Atos should use word embedding and bidirectional long short-term memory for classifying technical support tickets.

The literature review provided various models that are used for text classification. Support vector machines was reported to outperform similar machine learning models when class imbalance is present. This was less of an influence with neural networks. Two neural networks, deep neural network and recurrent neural network, suited the goal of this research. In total, three models with four architectures were compared, a support vector machines, two deep neural networks and a recurrent neural network. The input of those models were the different word representations. Two word representations were distinguished, document term matrix and word embedding. Support vector machines and deep neural network used the document term matrix, the other deep neural network and recurrent neural network made use of word embedding.

The architectures used for the models were:

- Linear support vector classifier for support vector machines, this is the classification model that uses the kernel linear for its algorithm.

- Multilayer perceptron for deep neural network, this architecture uses dense layers to create hidden layers.
- Bidirectional long short-term memory for recurrent neural network, this architecture solves the vanishing and exploding gradient problem that recurrent neural network has. The bidirectional part preserve information from both past and future.

The word representations used in feature engineering were:

- Term frequency-inverse document frequency for the document term matrix.
- Embedding layer before the models architecture for word embedding. Pretrained word embedding was reported to be insufficient in domain-specific tasks.

The text preprocessing was identical for each model. Since the data contained customer-specific and broad categories, relabeling of the categories was appropriate. As well as removing natural languages other than English. In spite of an apparent imbalance in the data, balancing the categories was not necessary. The literature described text preprocessing steps that were best suited for the scenario of domain-specific text classification; (1) removing characters, (2) tokenization, (3) lemmatisation, (3) removing stop-words, (4) thresholding word count, and (5) splitting data into training and test set. The last step, included defining a training size. A split of 85% training set and 15% test set was picked, to keep as much data for training.

Lastly, what is best for Atos, and therefore the steps that should be taken. There are uncertainties, caused by assumptions or being out of scope for this research i.e., (1) domain-specific problems, (2) written text long enough, (3) filtered out non-English tickets, and (4) enough data to compensate class imbalance. Still, this research shows that machine learning and neural networks models are very effective in classifying the category of tickets using their descriptive information. Getting the ideal text preprocessing, feature engineering and model take a lot of research and meanwhile having access to a working combination is valuable. Atos should implement the findings of this study and improve the individual components along the way. Prioritisation of the process is described in the next Chapter.

Chapter 9

Recommendations

As a follow-up to this research, several aspects were discussed that deserve attention. These aspects are:

(1) Better demarcation of the categories
The results debated that class imbalance and/or overlapping categories are a restraint for the model to get better weighted F_1 scores. If overlapping categories is the problem, adding more data does not increase the effectiveness of the model in classifying the categories of tickets. Hence, the overlap between categories needs to be addressed. Overlapping categories is fixed by putting a team of domain experts together to produce a set of labels that need to be predicted and map the existing categories to those labels. This resolves the problem for existing tickets. To prevent overlap of categories in future tickets, support agents need to be better informed or educated. Resolving overlap between categories, yields an effective prediction model with a weighted F_1 score of at least 90%. This assumption is made by looking at better defined categories i.e., 'eventManagement', 'userManagement', 'print', 'data' and 'hardware'.

(2) Keep the polylingual characteristic of ticket data
As described in the project context, just under 30% of the three available written fields are filtered out for being in a language other than English. To keep as much tickets as possible, the three fields are combined, so only ~8% of the tickets are removed. Research has shown that non-English words have little to no effect in the effectiveness of the model. Incorporating the polylingual characteristic of the ticket data is therefore advised. Although, an increase of the weighted F_1 score is not expected, it opens up the possibility to omit fields or to create a model that processes the three fields separately to predict the category. These changes have either a positive or negative effect on the effectiveness of classification. The data scientists in PXC need to execute and measure the effect of this change. After which, they report if the difference in operation time and effectiveness is beneficial enough to change the process.

(3) Calculate the business value
This study aims to measure the effect of text classification on ticket data, not the cost of running the model, which is imperative for Atos, since knowing which steps are cost-effective needs to be justified. Atos benefits from not only an effective model, but from a solution that has as little operation costs as possible.

If a step in the process takes thirty minutes and the effect on the model is nought point one percent, then it is likely that the step is not cost-effective. Getting this insight is achieved by logging the duration of each step and repetitively withholding one step and measuring its effect. After which, Atos is able to warrant each individual step. This process is best executed by a data engineer or data scientist.

(4) Research the effect of out-of-vocabulary
The assumption that ticket data contains domain-specific quirks, resulted in avoiding pretrained word embedding. However, the vocabulary is only 16.414 words, if only words are considered that are in more than 9 tickets and 169.469 if no limitations are set. It is likely that the word embedding comes across words that are out-of-vocabulary, when ticket data is used from a customer that was not present in the training process. The impact of using ticket data from an, for the model, unknown customer on the effectiveness of the model in classifying tickets needs to be measured. Anybody in the PXC team is qualified to perform this analysis, but it is best performed by the data scientists. They share the results with the team, so an appropriate decision is made. The result is either, that there is enough data to predict outside the known customers or there is not enough data for training an embedding which leads to switching to a pretrained word embedding.

(5) Research transformers and convolutional neural network
Convolutional neural network and transformers are effective models in text classification, trading blows with recurrent neural network, depending on the article. Considering that this study concludes that recurrent neural network is more effective in classifying the category of tickets, compared to less complex models (non-probabilistic binary linear classifier and feed forward networks). Research needs to be done on analysing those three architectures. Related works described the trend of using BERT in text classification, but also the potential bottleneck by combining the three fields. Implementing recommendation (2) clears that impediment. Although, it is not guaranteed that it delivers a positive effect on the classification, having a comparison between them leads to a more informative decision on which to use. This research is best left to interns or graduates, as it does not hinder Atos from implementing this research findings.

(6) Validation on a new customer
The validation is performed on a portion of the data of the same customers. Ideally, this is done on a new customer, since more insight is provided on the robustness of the models. Because, writing and slang could be different among customers. However, each label needs to be present, so the effectiveness of the model in categorising tickets can be measured among each class. Knowing that the model is an effective predictor for unseen customers is extremely important. Currently, it is unclear if the model can predict the category outside the scope of the customers that are in the training set. The client needs to provide ticket data with a correct category from a new customer, where the category contains all the trained classes. This needs to be executed by the data scientists of PXC.

References

Akkaradamrongrat, S., Kachamas, P., & Sinthupinyo, S. (2019). Text generation for imbalanced text classification. In *2019 16th international joint conference on computer science and software engineering (jcsse)* (p. 181-186). doi: 10.1109/JCSSE.2019.8864181

Bansal, S. (2018). A comprehensive guide to understand and implement text classification in python. *Analytics Vidhya*.

Barushka, A., & Hájek, P. (2019, 11). The effect of text preprocessing strategies on detecting fake consumer reviews. In (p. 13-17). doi: 10.1145/3383902.3383908

Camacho-Collados, J., & Pilevar, M. T. (2017, 07). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis..

Cherkassky, V., & Mulier, F. (1999). Vapnik-chervonenkis (vc) learning theory and its applications. *IEEE Transactions on Neural Networks and Learning Systems*, *10*(5), 985–987. doi: 10.1109/TNN.1999.788639

Dwarampudi, M., & Reddy, N. V. S. (2019). Effects of padding on lstms and cnns. *ArXiv*, *abs/1903.07288*.

Ellis, G., & Ellison, C. (2015). *Avoiding the 'watermelon' effect.* Retrieved from https://library.e.abb.com/public/afb41a2d4e6d45d9a47e2f71fc58521a/avoiding-the-%27watermelon%27-effect.pdf

Garette, D., Ljungörg, P., Nothman, J., Korobov, M., Bird, S., & Dimitriadis, A. (2021). Natural language toolkit. NLTK Project. Retrieved from http://www.nltk.org/nltk_data/

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Han, J., & Akbari, M. (2018, Apr.). Vertical domain text classification: Towards understanding it tickets using deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/11375

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1-12. Retrieved from https://doi.org/10.1021/ci0342472 (PMID: 14741005) doi: 10.1021/ci0342472

Hochreiter, S., & Schmidhuber, J. (1997, 11). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Hossin, M., & M.N, S. (2015, 03). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*, 01-11. doi: 10.5121/ijdkp.2015.5201

Joachims, T. (1998, 01). Text categorization with support vector machines. *Proc. European Conf. Machine Learning (ECML'98)*. doi: 10.17877/DE290R-5097

Jurafsky, D., & Martin, J. H. (n.d.). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.*

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on artificial intelligence - volume 2* (p. 1137–1143). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., & Barnes, L. E. (2017). Hdltex: Hierarchical deep learning for text classification. *CoRR*, *abs/1709.08267*. Retrieved from http://arxiv.org/abs/1709.08267

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, *10*(4). Retrieved from https://www.mdpi.com/2078-2489/10/4/150 doi: 10.3390/info10040150

Krouska, A., Troussas, C., & Virvou, M. (2016). The effect of preprocessing techniques on twitter sentiment analysis. In *2016 7th international conference on information, intelligence, systems applications (iisa)* (p. 1-5). doi: 10.1109/IISA.2016.7785373

Lai, S., Liu, K., He, S., & Zhao, J. (2016). How to generate a good word embedding. *IEEE Intelligent Systems*, *31*(6), 5-14. doi: 10.1109/MIS.2016.45

Leevy, J. L., Khoshgoftaar, T. M., Bauder, R. A., & Seliya, N. (2018, Nov 01). A survey on addressing high-class imbalance in big data. *Journal of Big Data*, *5*(1), 42. Retrieved from https://doi.org/10.1186/s40537-018-0151-6 doi: 10.1186/s40537-018-0151-6

Leopold, E., & Kindermann, J. (2002, Jan 01). Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, *46*(1), 423-444. Retrieved from https://doi.org/10.1023/A:1012491419635 doi: 10.1023/A:1012491419635

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 27). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the association for computational linguistics*, *3*, 211–225.

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th acm conference on recommender systems* (p. 59–66). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2959100.2959182 doi: 10.1145/2959100.2959182

Liu, A. Y.-c. (2004). *The effect of oversampling and undersampling on classifying imbalanced text datasets* (Unpublished doctoral dissertation). Citeseer.

Lyubinets, V., Boiko, T., & Nicholas, D. (2018). Automated labeling of bugs and tickets using attention-based mechanisms in recurrent neural networks. In *2018 ieee second international conference on data stream mining processing (dsmp)* (p. 271-275). doi: 10.1109/DSMP.2018.8478511

Mhaskar, H. N., & Micchelli, C. A. (1994). How to choose an activation function. In J. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems* (Vol. 6). Morgan-Kaufmann. Retrieved from https://proceedings.neurips.cc/paper/1993/file/51ef186e18dc00c2d31982567235c559-Paper.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, 01). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, *2013*.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021, apr). Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, *54*(3). Retrieved from https://doi.org/10.1145/3439726 doi: 10.1145/3439726

Moss, H. B., Leslie, D. S., & Rayson, P. (2018). *Using j-k fold cross validation to reduce variance when tuning nlp models.*

Noble, W. S. (2006, Dec 01). What is a support vector machine? *Nature Biotechnology*, *24*(12), 1565-1567. Retrieved from https://doi.org/10.1038/nbt1206-1565 doi: 10.1038/nbt1206-1565

Padurariu, C., & Breaban, M. E. (2019). Dealing with data imbalance in text classification. *Procedia Computer Science*, *159*, 736-745. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050919314152 (Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019) doi: https://doi.org/10.1016/j.procs.2019.09.229

Poolsawad, N., Kambhampati, C., & Cleland, J. (2014, 07). Balancing class for performance of classification with a clinical dataset. *Lecture Notes in Engineering and Computer Science*, *1*, 237-242.

Revina, A., Buza, K., & Meister, V. (2020, 01). It ticket classification: The simpler, the better. *IEEE Access*, *8*, 193380-193395. doi: 10.1109/ACCESS.2020.3032840

Salcianu, A., Golding, A., Bakalov, A., Alberti, C., Andor, D., Weiss, D., ... Koo, T. (2020). Compact language detector v3. Google Inc. Retrieved from https://github.com/google/cld3

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). The performance of lstm and bilstm in forecasting time series. In *2019 ieee international conference on big data (big data)* (p. 3285-3292). doi: 10.1109/BigData47090.2019.9005997

Sutskever, I., Martens, J., & Hinton, G. (2011, 01). Generating text with recurrent neural networks. In (p. 1017-1024).

Telnoni, P., Budiawan, R., & Qana'a, M. (2019, 11). Comparison of machine learning classification method on text-based case in twitter. In (p. 1-5). doi: 10.1109/ICISS48059.2019.8969850

Vijayarani, S., Ilamathi, J., & Nithya, S. (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7–16.

Wahba, Y., Madhavji, N. H., & Steinbacher, J. (2020). Evaluating the effectiveness of static word embeddings on the classification of it support tickets. In (p. 198–206).

Wang, J., & Zhang, M.-L. (2018). Towards mitigating the class-imbalance problem for partial label learning. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (p. 2427–2436). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3219819.3220008 doi: 10.1145/3219819.3220008

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding data augmentation for classification: When to warp? In *2016 international conference*

on digital image computing: Techniques and applications (dicta) (p. 1-6). doi: 10.1109/DICTA.2016.7797091

Yanminsun, Wong, A., & Kamel, M. S. (2011, 11). Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*. doi: 10.1142/S0218001409007326

Zemp, M. (2021, 01). *Text classification of service desk tickets.*

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010, Dec 01). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1), 43-52. Retrieved from `https://doi.org/10.1007/s13042-010-0001-0` doi: 10.1007/s13042-010-0001-0

Zhou, Z.-H., & Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, *18*(1), 63-77. doi: 10.1109/TKDE.2006.17

Appendices

Appendix A

Classification Report

Weighted f1-score for SVM: 0.8359177396298049

	precision	recall	f1-score	support
applications	0.81	0.85	0.83	8777
data	0.93	0.85	0.89	902
eventManagement	0.92	0.98	0.95	2812
hardware	0.90	0.90	0.90	1793
hardwarePC	0.67	0.60	0.63	1911
network	0.66	0.60	0.63	931
outlookSkype	0.60	0.60	0.60	694
passwordReset	0.81	0.85	0.83	1961
print	0.84	0.85	0.85	1035
software	0.85	0.84	0.84	10642
userManagement	0.94	0.88	0.91	4957
voiceVideoMobility	0.69	0.68	0.69	931
accuracy			0.84	37346
macro avg	0.80	0.79	0.80	37346
weighted avg	0.84	0.84	0.84	37346

Figure A.1: Classification report of LinearSVC showing precision, recall f_1-score and support for all 12 classes

Weighted f1-score for DNN: 0.8174864070520869

	precision	recall	f1-score	support
applications	0.75	0.87	0.80	8777
data	0.96	0.83	0.89	902
eventManagement	1.00	0.98	0.99	2812
hardware	0.93	0.86	0.89	1793
hardwarePC	0.68	0.50	0.58	1911
network	0.84	0.39	0.53	931
outlookSkype	0.72	0.42	0.53	694
passwordReset	0.80	0.86	0.83	1961
print	0.95	0.72	0.82	1035
software	0.78	0.87	0.82	10642
userManagement	0.98	0.85	0.91	4957
voiceVideoMobility	0.77	0.54	0.63	931
accuracy			0.82	37346
macro avg	0.85	0.72	0.77	37346
weighted avg	0.83	0.82	0.82	37346

Figure A.2: Classification report of Multi Layer Perceptron (MLP) with TFIDF showing precision, recall f_1-score and support for all 12 classes

Weighted f1-score for DNN: 0.8311399039836819

	precision	recall	f1-score	support
applications	0.80	0.85	0.82	8777
data	0.91	0.82	0.86	902
eventManagement	1.00	0.98	0.99	2812
hardware	0.90	0.87	0.89	1793
hardwarePC	0.69	0.59	0.63	1911
network	0.72	0.59	0.65	931
outlookSkype	0.66	0.59	0.62	694
passwordReset	0.74	0.87	0.80	1961
print	0.88	0.87	0.88	1035
software	0.82	0.86	0.84	10642
userManagement	0.96	0.83	0.89	4957
voiceVideoMobility	0.72	0.63	0.67	931
accuracy			0.83	37346
macro avg	0.82	0.78	0.79	37346
weighted avg	0.83	0.83	0.83	37346

Figure A.3: Classification report of Multi Layer Perceptron (MLP) with word embedding showing precision, recall f_1-score and support for all 12 classes

Appendix B

Model's Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 256, 256)          64000000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 256, 256)          0
_________________________________________________________________
bidirectional (Bidirectional (None, 256, 512)          1050624
_________________________________________________________________
global_max_pooling1d (Global (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 12)                6156
=================================================================
Total params: 65,056,780
Trainable params: 65,056,780
Non-trainable params: 0
_________________________________________________________________
```

Figure B.1: Summary of the RNN model, consisting of an embedding + BiLSTM + output layer

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 256, 256)          64000000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 256, 256)          0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 512)               33554944
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                6156
=================================================================
Total params: 97,823,756
Trainable params: 97,823,756
Non-trainable params: 0
_________________________________________________________________
```

Figure B.2: Summary of the DNN model, consisting of an embedding + 2 hidden + output layer

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               321332736
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                6156
=================================================================
Total params: 321,601,548
Trainable params: 321,601,548
Non-trainable params: 0
_________________________________________________________________
```

Figure B.3: Summary of the DNN model, consisting of 2 hidden + output layer

Glossary

B		
	BERT	bi-directional encoder representation from transformers
	BiLSTM	bi-directional long short-term memory
C		
	ConvNet	convolutional neural network
D		
	DNN	deep neural network
	DTM	document term matrix
E		
	EX	employee experience
H		
	Hyperparameter	parameter to control the learning process
L		
	LSTM	long short-term memory
M		
	MLP	multilayer perceptron
N		
	NLP	natural language processing
P		
	PXC	proactive experience center
	PMI	pointwise mutual information
R		
	RNN	recurrent neural network
S		
	SVM	support vector machines
	SVC	support vector classifier
	SLA	service level agreement
	SVD	singular value decomposition
T		
	TFIDF	term frequency-inverse document frequency
X		
	XLA	experience level agreement