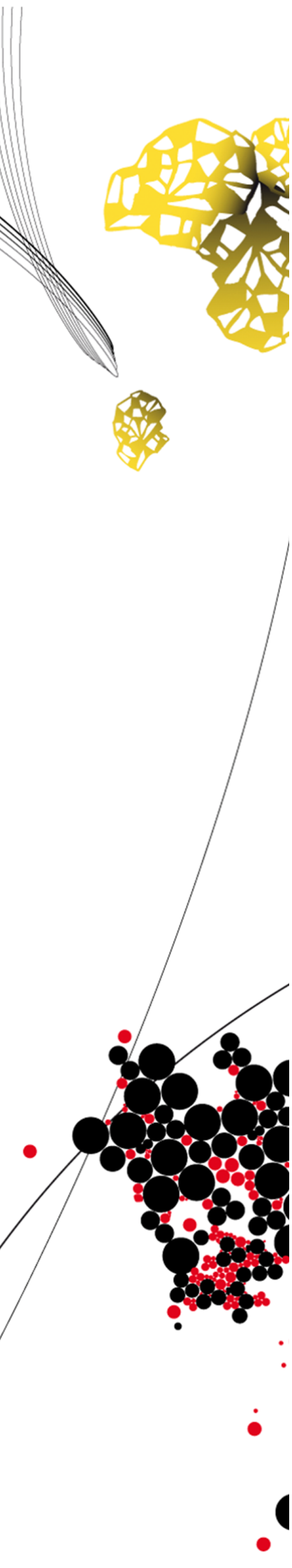


UNIVERSITY OF TWENTE.

**Faculty of Engineering,
Mathematics and
Computer Science**

Classification-based approach for Question Answering Systems: Design and Application in HR operations

Levi van der Heijden

September 2019

University Supervisors:
Dr. C. Seifert (EEMCS-DS)
Prof. dr. T. Bondarouk (BMS-HRM)

External Supervisors
A. Feijen MA.
G.G.M. Erdkamp MSc, MA.

Business Information Technology
Faculty of Engineering,
Mathematics and
Computer Science
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

Abstract

In this thesis, a Proof of Concept for an automated Question Answering (QA) System to answer HR-related questions is developed and evaluated. This is done by classifying employee questions into question categories for which standard responses can be sent to employees. Several Classification methods are evaluated, under which Support Vector Machine, RandomForest, XGBoost and a Bi-LSTM Neural Net. Moreover, several methods for text cleaning, label discovery and text transformation are used and evaluated to operationalize the unlabeled client dataset. A desirable micro average precision of 89% and recall of 81% is achieved with the final Proof of Concept. Moreover, a self-learning Cloud-based Solution was designed within the client context in which the Proof of Concept can be deployed. Overall, this study provides evidence for the potential impact of Artificial Intelligence on HR in terms of operational and strategic value, as well as guidance into what is required to achieve this value.

1 Introduction

This section introduces the societal and business problems addressed by this solution. From the resulting research problem and stakeholder requirements, several research questions are formulated that were tackled by this design study. This is followed by the research methodology, which dictates the structure of this thesis.

1.1 Research Problem

Organisations are always looking for new opportunities to create unique value that separates them from their competitors. In the current 'fourth industrial revolution' or 'industry 4.0', where the digital and the physical are integrated, an organisation can create value like never before through new business models leveraging the latest technological developments [1, 2]. As a concept that emerged from the manufacturing domain [2, 3], the supporting technologies driving the industry 4.0 could also be applied to service domains, such as Human Resource Management (HRM) [4]. In this study, a self-learning machine learning system is designed to answer employee questions related to HR practices and policies in a fast and standardised manner. Here, two technologies that support the industry 4.0 [1], are applied to create value in a service-oriented manner within the HR domain. These are *Artificial Intelligence (AI)* and *Cloud Technologies*.

Thus far, HRM and Industry 4.0 were addressed mostly in the context of learning and development. HRM practices that enable the rapid building of capabilities required for industry 4.0 are needed [5]. In this study, this logic is turned around: HRM is not used to enable the use of industry 4.0 supporting technologies, but these technologies are applied to create a solution enabling HRM. Applying technologies supporting the industry 4.0 to the HR domain is a novel and interesting endeavour, as the field of HRM has been criticised lately on the relative weak adoption and application of data analytics, one of the supporting technologies this study will address [6]. A way to improve the state of data analytics with HRM is to provide empirical evidence and more practical studies, as opposed to the abundant high-level studies within the field of HR analytics [7]. HR analytics is defined as "the systematic identification and quantification of the people drivers of business outcomes, to make better decisions" [8]. This study contributes to the field of HR analytics by providing a case study of the implementation of a solution that utilises employee data to develop an AI to make automated decisions in a cloud solution, driving new ways of value creation in the vain of the industry 4.0.

The context of the case study is the operations of the HR department of *VodafoneZiggo*, specifically within the team *HelloHR*. VodafoneZiggo is a Dutch telecom organisation operating with 7438 employees [9]. Their core business is the provision of cable television, internet and phone services to both residential and commercial customers, with the purpose of *'enjoyment and progress with every connection'*

[10]. VodafoneZiggo currently connects 7.2 million households, covering 91.6% of households in the Netherlands [11]. Furthermore, VodafoneZiggo is a merger of the subsidiaries of two large international organisations: Vodafone of Vodafone Group and Ziggo of Liberty Global, both having a 50:50 stake in VodafoneZiggo [12]. Since the merger, the goal of the HR department has been making this joint venture a success, by developing a shared culture and transparent organisational structure [9].

Within the HelloHR team, the main goal is enabling the HR practices and policies within the organisation by helping employees, managers and business partners when they have HR-related issues. They do this by answering questions, mutating employee files, creating knowledge documents and supporting internal communications. At this time, the department's workload is high, increasing the time employees have to wait on their answers or changes, which affects the employee experience and the ability of employees to do their job. To decrease the workload of HelloHR employees and improve the HR service delivery, which improves the employee experience and decreases delays in work due to HR issues, VodafoneZiggo wants to tackle the high volume of employee questions, averaging about 4500+ tickets a month, visualised in figure 1. Here, a ticket is an employee question registered in the SaaS solution 'TOPdesk', in which HelloHR manages and answers employees issues.

To alleviate the high workload of HelloHR and improve the HR service delivery, a question answering system has been developed. This question answering system consists of an AI able to classify the question, and a cloud solution that, using the classification of the AI, can answer the employee questions. During this study, the generalised research problem was:

Improve the HR services of the HR department by designing a question answering system supported by machine learning in the cloud to reduce the response time on employee questions and amount of employee questions which will reduce the operational workload and improve HR service delivery

The resulting Proof of Concept (POC) can answer 7.2% of the e-mail based question workload of HelloHR, with the potential to scale up. With the current solution, only 13% of questions are wrongly answered. The scalability of the solution and manual maintenance are simplified as the solution supports automated learning in the cloud using human corrected labels whilst keeping the disruption of the existing question answering process by HelloHR at a minimum. To achieve this, a mix of methods was used, involving information retrieval, semi-supervised- and supervised machine learning. The cloud solution used was Amazon Web Services. Moreover, the methods and techniques used to design and develop this solution are easily transferable to other contexts. Finally, several recent innovations in machine learning are described that can improve further iterations of this PoC.

In sum, this study contributes an extensive case study regarding the use of industry 4.0 technologies within HRM, aiding both applied machine learning engineers and HR practitioners to identify and develop new use cases.

1.2 Requirements & Scope

Before a question answering systems supported by machine learning in the cloud was selected as the solution scope, a VodafoneZiggo specific research problem was formulated in search of potential other interventions that could lead to a reduction in the workload of HelloHR. The initial goal of VodafoneZiggo was to "*Decrease the workload of HelloHR by reducing the number of employee questions*". The expectations for this design study was the design of a Proof of Concept as a start towards this goal. Here, the goal of the Proof of Concept was to answer the 'top 7 most asked questions by HelloHR'. Whilst several different interventions could achieve the same goal, for example by changing communications or HR policies and practices, the final solution was scoped down to technological intervention. For this, several functional and non-functional requirements were identified.

Here, the *functional requirements* of the solution were:

- The solution should be able to **identify** what question is asked by the employee
- The solution should be able to **respond** to the question of the employee

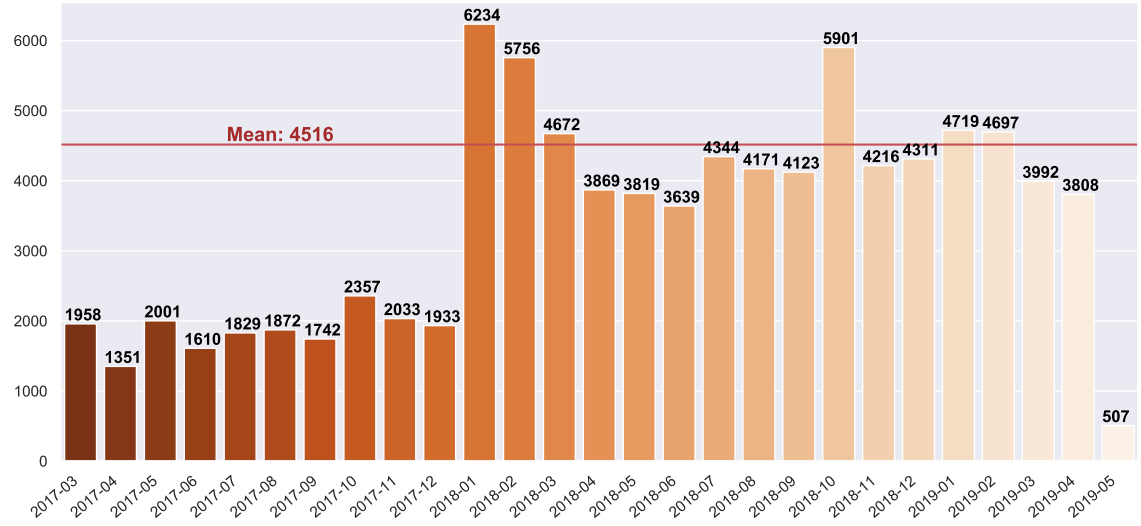
Considering these two functional requirements, two potential technological solutions were identified by VodafoneZiggo: a chatbot or an automatic e-mail answering system proposed initially by a consulting firm, which was rescoped as a question answering system in this thesis.

A *chatbot* can be described as "a virtual agent that serves humans as a natural language user interface with data and service providers" [13]. These are interactive solutions that have recently gained renewed traction due to advancements in chatbot technologies and the shift in preference towards real-time messaging as a channel to communicate with businesses [14, 15]. Chatbot solutions are made to function in dialogue, focusing on multiple exchanges to help solve issues, and can function on both text and speech. These can be either so-called dialogue managers, where the chatbot responds based on a humanly designed way to certain intents defined by humans during the design [16, 17, 18]. Other chatbots generate answers using machine learning techniques such as sequence2sequence, where an answer is generated based on the question [13]. These chatbots are however unreliable for now and unsuitable for business environments, such as Tay by Microsoft has shown [19].

A *Question answering (QA) system* is a broad term for a system able to analyse a textual string and provide an answer based on the text [20, 21]. Various types of QA systems exist borrowing techniques from the field of information retrieval, information extraction and machine learning [20, 22]. Where chatbots are a solution made to have a dialogue with a human, QA systems are a solution made to answer a question. In this design study, the term QA system is scoped down to refer to a system able to respond to a *single* question with an answer.

As both solution domains fulfil the functional requirements, *non-functional requirements* were formulated by VodafoneZiggo determined to select one solution domain and evaluate the Proof of Concept based on the scope of answering the top 7 questions of employees to HelloHR.

Figure 1: *The amount of employee tickets in TOPdesk over time. The mean calculation excludes the full year 2017, as the platform was Ziggo only till 2018, and May 2019, as the data ranges from the* 2^{nd} *of March 2017 till the* 5^{th} *of May 2019.*



- **Precision**: The solution should prevent returning a wrong response, or a 'false positive'.
- **Impact**: The solution should recognise questions that are in scope, thus not miss questions it should be able to answer.
- **Cost-effectiveness**: There should be a realistic time frame for the return on investment.
- **Maintainability**: The manual effort to maintain the quality of the solution should be minimal.
- **Scalability**: The effort to upscale the scope of the solution should be minimal.
- **Fault-tolerant**: The system should not disturb any existing processes on a technical failure, meaning system outage or errors.

It is difficult to assess which solution to choose based on the first two criteria; this can only be evaluated afterwards. However, the suitability of the other four requirements can be evaluated upfront.

The dialogue managers, chatbots most suitable for businesses, are scalable but require expensive solutions to design the dialogue managers in, which require a team of experts that know what employees ask and how to respond properly. Expanding the dialogue manager also requires an extra team that creates the new dialogues, and changes in how the business operates might require changes to the existing dialogue, which causes more maintenance. Last, fault-tolerance is high, as the solution will not exist in any existing processes.

QA systems utilising machine learning can be designed in such a way to learn new situations automatically, decreasing required maintenance. Moreover, development costs are relatively low, as one only needs a dataset, and not a team of experts, to get started with developing the QA system. Finally, due to the low maintenance cost, it is expected that a QA system over time, for the small scope of simple questions, will result in a more cost-effective than a full chatbot solution.

A final element impacting the solution scope is the soft requirement to make the QA system run in a Cloud environment as opposed to a local environment. This soft requirement was introduced at an early stage of the study and emerged as part of the strategic move of VodafoneZiggo to use this design study as a proof of concept for automated machine learning in the cloud.

In sum, the scope of the solution design was limited to a *question answering system supported by machine learning in the cloud.*

1.3 Research Questions

Based on the initial research problem, the following technical research problem is formulated as the question:

How to design a question answering system that responds to operational questions at the HR department which will reduce the workload and response time to improve the operations of the HR department?

To answer this question, several smaller questions have to be answered. These are partly based on the non-functional requirements for this design study and are aimed at evaluating the final solution. The other part of the research problem concerns the development of the solution design. This will involve a dataset which has not been labelled for question answering, on which machine learning techniques will be applied to answer employee questions. Moreover, the resulting machine learning techniques have to be trained automatically and deployed in the cloud, and interface with the existing TOPdesk solution in VodafoneZiggo. This brings the following set of research questions.

1. What is a good question-answering process involving machine learning?
   (a) What machine learning tasks are involved?
   (b) What machine learning models exist to perform these tasks?

(c) How can these models be used in practice?

(d) What is needed to operationalise these models?

2. What are the costs and benefits of the resulting solution?

(a) What is the performance of this solution?

(b) What is the effect of the solution on the operations of the HR department?

(c) What is the expected timeframe for the return on investment of the solution?

1.4 Research Methodology

Throughout this study, the design science methodology for information systems and software engineering by Wieringa [23] has been applied. Within the methodology of Wieringa [23], the engineering cycle describes a rational problem-solving process that involves the following tasks:

1. *Problem investigation*: What phenomena must be improved and why? During this phase, the stakeholders and their goals are defined, as well as a conceptual problem framework in which the problem phenomena, it's causes, mechanisms and reasons are described. Moreover, the contribution of a potential solution to the effects of the problem phenomena is explained.

2. *Treatment design*: In this phase, one or more artefacts are designed that could treat the problem. Requirements for this artefact are specified and their contribution to the stakeholder goals should be clear. Several treatments might be available to treat the problem. If these are not sufficient or available, new ones must be designed.

3. *Treatment validation*: Would these designs treat the problem? Here, the designed artefact is tested to see if it is sufficient to treat the problem. This is done by analysing the effects of the design on the problem, analyse the trade-offs compared to different artefacts, analyse the sensitivity to different contexts and analyse if the effects satisfy the requirements to contribute to the stakeholder goals.

4. *Treatment implementation*: The resulting artefact is implemented in the problem context to treat the problem.

5. *Treatment evaluation*: How successful has the treatment been? Similar questions are asked as during the problem investigation phase, and if it appears that the treatment does not sufficiently solve the problem, a new engineering cycle is started beginning with the problem investigation.

This study was performed as Technical action research (TAR), where the use of an experimental artefact helps a client and provides leanings about its effects in practice [23, 24]. The core of this type of research is that the researcher plays three different roles:

1. *Technical researcher*, who develops an artefact to help the clients situation. In this case, a question answering system supported by machine learning in the cloud is designed and tested as a PoC to solve the issue of large volumes of employee questions.

2. *Emperical researcher*, who tries to validate knowledge questions about the treatment. In this scenario, this is the validation of several machine learning models in this contest to discover the best.

3. *Helper*, who tries to apply the use case to fit the client's situation. This is the application of the system in the specific cloud environment of the client, in this case, TOPdesk and Amazon Web Services, and application to the specific questions within the client's organisation.

Key is to keep these roles conceptually separate [23, 24]. Thus far, this has been done by first describing the problem in a manner that applies to a general situation, such as questions entering an HR department. Then the situation is further

specified to the client's context in which the artefact will be designed. Afterwards, an analysis is done to evaluate how the results of the artefact in the context of the client could be related to a more general context.

A large part of a standard TAR was executed in this thesis, but this stops at the point of implementing the solution at the client-side beyond a testing environment, as this was out of scope of the study, considering a Proof of Concept was required by the client and provides enough insights into the potential of Industry 4.0 supporting technologies in the context of HRM. All in all, this study is structured as followed.

Section 3 'Background' will contain more information about the context of the client situation. Furthermore, the core elements of the QA system, machine learning and cloud technologies, are further described to discover what machine learning tasks are involved in such system, what machine learning models exist, how these can be deployed in practice in a cloud environment. Moreover, the evaluation metrics and validation methods used in this thesis to evaluate the machine learning models are described. This section will conclude with a high-level design of a QA system supported by machine learning in the cloud.

Based on section 3 'Background', section 4 'Solution Design' will introduce the application of the high-level solution design to the client context. Here, the end-to-end functionality will be described of an ideal solution which involves a self-learning QA system in Amazon Web Services that is connected to the SaaS solution used by HR employees at the client. Section 5 'Proof of Concept' will follow up on this by introducing the steps taken during this thesis to aid in the realisation of the solution for the client. First, the operationalisation of the client dataset is described, involving the creation of labels in the unlabelled client dataset. Moreover, a proof of concept design is introduced in which the contribution of this thesis to the client solution is described.

In section 6 'Experimental Setup', methods are described to create and validate the machine learning models enabling the QA system. The methods to test the performance of various machine learning pipelines is discussed. Two experiments are described that will reveal the best model configuration and if more data can improve these models.

In section 7 'Results' the several configurations of the POC are then validated using the methods and experiments from section 6 'Experimental Setup', testing the trade-offs of different configurations of the POC. The outcomes of the execution of the methods from the approach section are presented, from which a final, best POC artefact is created for the client. This followed by section 8 'Discussion', in which these results are evaluated in terms of the impact on the problem phenomenon and client requirements, as well as the generalisation to other contexts.

This thesis concludes with section 9 'Conclusion & Future Outlook', where a summary of the results, the impact this has on the domains of applied machine learning and HR analytics, and a future outlook describing future experiments that could lead to an improvement of this artefact is described, which involves methods that were not in scope or feasible for the PoC of this thesis.

2 Related Work

Question Answering systems have a long history, dating back to 1961 with BASEBALL, a question answering system able to answer basic questions about baseball [22, 25]. It did this by transforming a question into a query that could be used to extract information from a database. Now QA systems have evolved, where the stream of factoid question answering dominates; given a description of an entity, identify what is discussed in a question [26].

In general, QA systems are built to analyse a question, retrieve documents relevant to the question, extract an answer from these document and generate a response [22]. To do this, techniques from information retrieval and information extraction have been applied [22], and recently also the use of neural networks [26].

Currently, the Stanford Question Answering Dataset (SQuAD) is used to benchmark QA systems [27]. The QA systems that score the highest on the SQuAD dataset, use a language model

named *BERT* in combination with an ensamble of other techniques, where BERT is a language model based on sequences of text that learns word representations depending on the context in which words are embedded [28].

These QA systems are work with a large set of documents to retrieve answers from. In the scope of this study however, these documents are absent. Therefore, the QA system in this thesis was limited to question analysis. Here, question analysis is treated as a classification problem. Text classification in general uses similar techniques as mentioned above, from information retrieval, information extraction and machine learning.

Various methods for feature extraction and machine learning have been utilised and evaluated in the context of question answering systems [29]. These can involve bag-of-words models [29], where each sentence is represented as a 'bag' of words without a specific sequence and sequence-based models [28, 30], where the sequence of words in a sentence is preserved.

An issue within this field is that no perfect combination of the method of feature extraction and machine learning model can be determined a priori for a dataset [29]. Therefore, a subset of methods is selected to explore based on previous studies in QA systems and text classification to limit the scope of methods to explore. During this study, methods from both text classification as well as conventional QA systems are used to built a simple QA system focusing on the task of question extraction through classification. For feature extraction, methods from the fields of word embeddings, from which BERT originates, are explored, as well as term weighting, a technique from information retrieval. For machine learning, bag-of-words [29] and sequence-based [31] machine learning models are explored.

3 Background

In this section, the situation of the client is described, as well as the high-level design of the QA system supported by machine learning in the cloud that will alleviate the client situation. This is done by first describing the client situation in terms of the problem process, in which all systems and elements relevant to the problem phenomena are described, as well as the intervention point of a possible solution. Afterwards, the core elements of the QA system are described in the form of a short literature review to explore various machine learning models to aid the client and to describe the various elements of the Amazon Web Services solution that are required to use these machine learning models in practice. This section closes with the description of a high-level design that was implemented and tested using historical data of the client, of which the results are described in the result section.

3.1 Client Situation

HelloHR, the client, operates within the HR department of VodafoneZiggo. One of their tasks is answering questions of employees, however, due to the high workload, this process isn't executed as desired. Employees have to wait long times for responses, and the HelloHR employees feel overloaded. Within this context, the QA system will alleviate the workload and improve the HR service delivery towards the employee by providing lower lead times for question answering.

Employee issues arrive at HelloHR in various ways. One of these ways is by *e-mail*, compromising about 70% of the total ticket volume within TOPdesk (see figure 2). Other channels involve:

- *Word-of-mouth*, i.e. an employee walking past the desk of an HelloHR employee
- *Skype for Business*, the instant messaging tool developed by Microsoft [32]
- *Yammer*, the enterprise social network platform developed by Microsoft [33]
- *Phone*, for example an employee calling to ask a question.

When questions arrive through these other channels, an employee has to manually log a ticket in TOPdesk, in contrast to e-mail, where every e-mail enters the TOPdesk platform by default. This makes it difficult to assess the

actual question volume that arrives by e-mail. Moreover, follow-up on asked questions can occur through different channels than the initial question was asked. Employee issues can be one of three kinds: a question, an incident or a request. The definitions of these categories are too ambiguous for the QA system; anything that can be answered with a standard e-mail is something the QA system in the current scope can solve. Executing processes or changes in various systems also referred to as mutations, are out of the scope of a QA system by definition. E-mails are estimated to require about 5 minutes on average for cases with low to medium difficulty, which is within the scope of the QA system for the PoC. Extrapolating this to the mean ticket load of 3145, about 2 Full-Time Equivalents (FTE), making the QA system equivalent to two full-time employees. For the client, a worthwhile opportunity, especially since the QA system is available at all times and replies almost instantly.

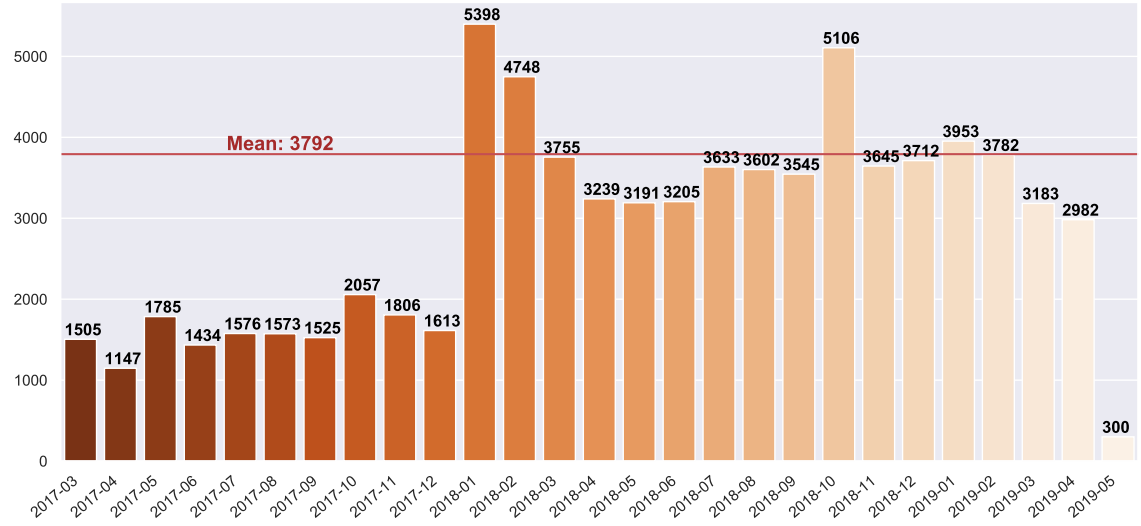
TOPdesk is the SaaS solution that enables the answering, logging and audit of questions arriving at HelloHR and answers given. TOPdesk has an extensive Application Programming Interface (API) that allows web services to make changes to various elements of tickets [34]. Moreover, TOPdesk has an Event & Action Management Module, which allows for the automatic scheduling of tasks such as changes to the content on TOPdesk or exchange of information with web services depending based on specific events [35].

Aside from the employee *request* and the HelloHR *action*, TOPdesk stores various fields, of which the following are of the importance of this thesis:

- *Subject*, containing the subject of the incoming e-mail
- *Category*, containing one of 23 categories that have to be filled in before the employee sends a response.
- *Subcategory*, containing one of 147 subcategories. Only some can be selected based on the category selected by the employee. This field is also mandatory to fill in before sending a response
- *TicketID*, a unique number for the ticket

With exception of the action, subject and request fields, each element in the ticket contains a

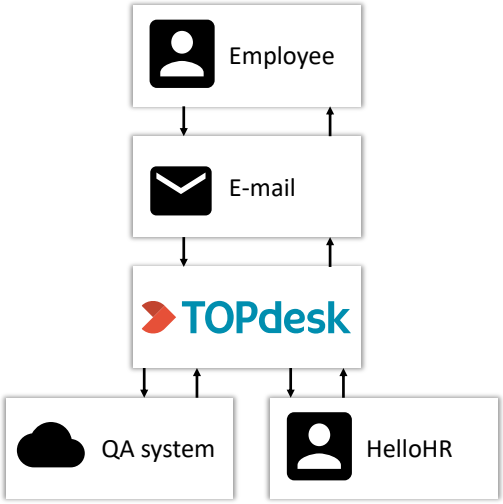
Figure 2: *The amount of employee tickets in TOPdesk over time arriving by mail. The mean is calculated the same as in figure 1.*



Unique Identifier (UnId), which ensures that in the back-end of TOPdesk, nothing changes if the name of a category or subcategory or other field changes.

After the ticket has been received, the category, subcategory and action have been filled in, and the ticket is either put on 'to be checked' or 'closed', a response is sent to the employee. If an employee then replies by e-mail with the ticketID in the subject header, the request of the employee is stored under the same ticketID, creating a chain of requests and actions.

Figure 3: *The client context in which the QA system will exist.*



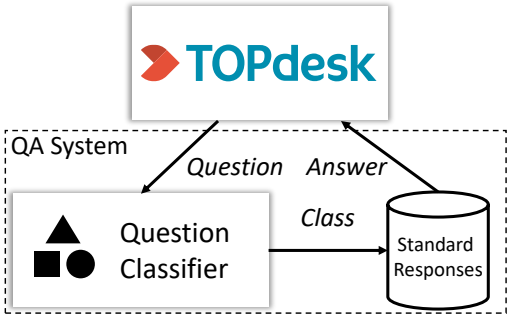
Generalising this to a QA system, the textual string would consist of a combination of the Subject and the Request, and the Action is something the QA system should provide. If the QA system is not able to provide an answer, because the question is out of scope or because the system is simply not certain enough about the incoming e-mail, the original business process should be followed, where an HelloHR employee can answer an incoming employee question. The process, or context, the QA system will exist in is visualised in figure 3.

3.2 QA Systems

QA systems usually exist out of three main tasks: question analysis, answer extraction and response generation [22, 36]. Most QA systems are targeted at the extraction of answers from knowledge bases using the question of a user. In this design, however, the answer extraction and response generation are simplified, as the question scope is quite limited.

Within the scope of this study, questions arrive in the form of an e-mail. As such, a question consists of the body of the e-mail and the subject of the e-mail. Based on this, the question analysis process extracts a class from the body and subject of the e-mail, based on which an answer can be retrieved from a database, which is returned to the system or person asking the question. In sum, the question analysis process is seen as a classification process, resulting in a heavy simplification of the answer retrieval and generation elements of the QA system. A simplified design of this QA system can be found in figure 4.

Figure 4: *The core components of the QA system.*



The problem to solve for the QA system to work is a text classification problem, where the question classifier has to accurately classify questions into multiple predefined classes. In this section methods from the fields of information retrieval and machine learning are evaluated that can be used for text classification. These methods are divided and discussed in three different topics.

First, methods that transform the text into numerical features are discussed. Second, sev-

eral supervised machine learning models are discussed that utilise numerical features in combination with labels to learn an optimal classification boundary for these numerical features. Third, as the working dataset is expected to have a large difference in support from the various question classes that it should identify, the topic of class imbalance is introduced, as well as methods to overcome this imbalance. An overview of methods that will be introduced in the following subsections can be found in table 1.

Finally, several evaluation metrics commonly used in machine learning are introduced. These can be used to evaluate and select a model that best fits the client dataset in order to build the best QA system with the set of identified transformation and classification methods.

Transform	**Classify**	**Imbalance**
Frequency	SVM	Weighted
TFIDF	RF	Undersampling
Word2Vec	GBT	Oversampling
FastText	Bi-LSTM	
ELMo		
BERT		

Table 1: *Overview of methods which will be discussed in this background study.*

3.2.1 Transformation

In order for text to be interpretable for a machine, a numerical transformation has to occur that encodes this text into an input vector of numbers. To do this, various transformation methods exist that extract numerical features from text in a different way, impacting the way machine learning models can extract classification boundaries from this text. In this section, several transformation methods are discussed, ranging from quite simple methods of encoding to state of the art methods.

Term Frequency

Term Frequency vectorization involves the mapping of each word in a document to a sparse matrix, where the columns of this matrix represent a single word in the vocabulary of a corpus of documents. It uses the Bag-of-Words concept, where each document is represented as a list of words without preserving order. The size of the frequency matrix will be equivalent to the vocabulary size of the input dataset for the Count Vectorization. When a document is transformed using this method, the occurrence of a word is counted by adding 1 to the column index in the created sparse matrix for a specific word for each time it occurs in the document. Whilst this method is fairly straightforward, the size of the frequency matrix can become quite large for bigger datasets, resulting in a large number of features. Moreover, in this method, each word is equal in importance after the transformation.

Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) is a slightly more sophisticated method compared to TF. Whilst fairly simple and quite old, this technique improves on flat TF by adding a score on how relevant a certain term is in a document, depending on the input collection of documents [37]. In general, the application of TF-IDF is as follows: Given a collection of documents D, a single word w and an individual document $d \in D$, one calculates, for each word:

$$w_d = f_{w,d} \times log(|D| f_{w,D}) \quad (1)$$

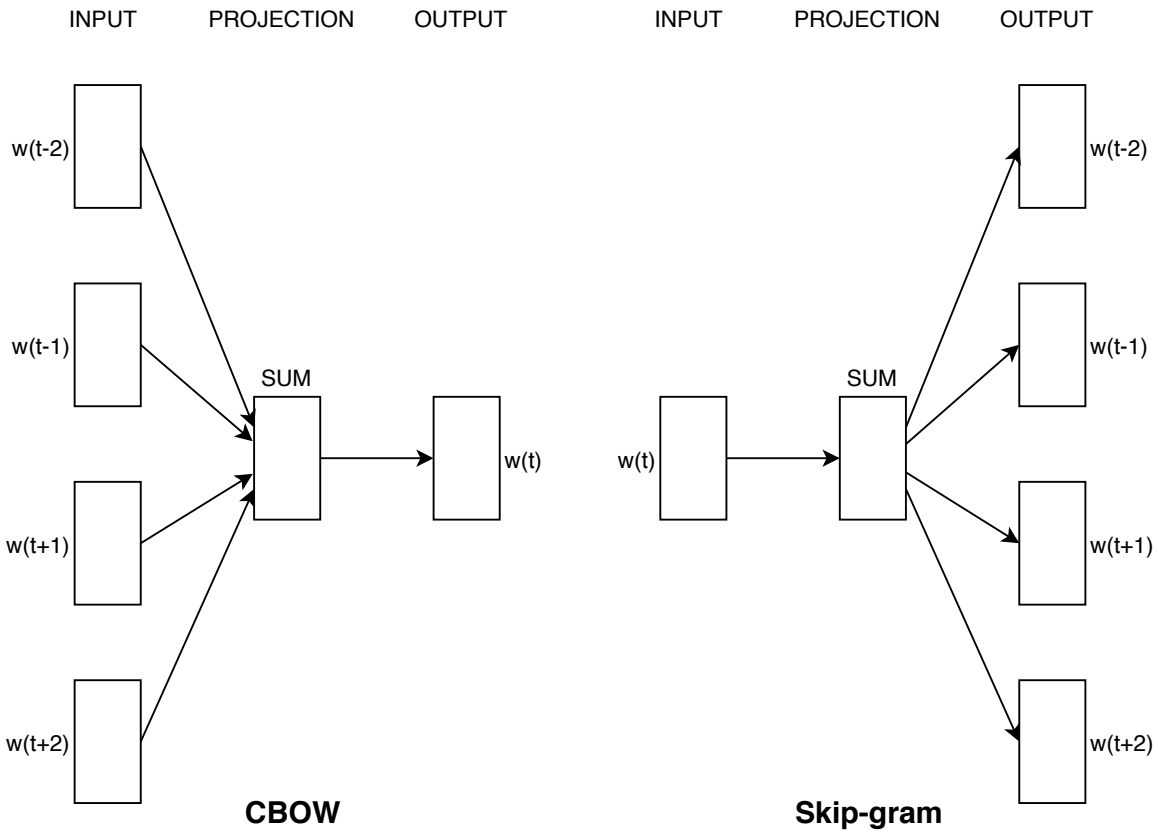
Here, $f_{w,d}$ describes the frequency of a word w in document d compared to other words, $f_{w,D}$ describes the total amount of documents the word is in and D describes the total frequency of documents [37].

The transformation using TF-IDF is similar to that of TF. a sparse matrix is created, but instead of flat counts, the frequency of a word in a document is multiplied with its TF-IDF score. Whilst these methods transform basic textual features into a numeric encoding with quite simple logic, they fail to capture more complex elements of language into this encoding, such as the similarity between words.

Word2Vec

A method to capture the similarity between words in the encoding of text is *Word2Vec*.

Figure 5: *The CBOW and Skip-gram architectures for word embeddings as visualised by Mikolov [38, 39]. Here $w(t)$ represents a word and its relative position to the word that should be predicted. The projection layer, also referred to as the hidden layer, learns the word embeddings.*



Here, one trains a shallow neural network of one of two architectures: Continuous Bag-of-Words (CBOW) or Continuous Skip-Gram [38, 39]. In the CBOW architecture, one trains a shallow neural network with one hidden layer of which the input is a window of words around a word in a document and an output softmax layer with the size of the vocabulary of the corpus. The goal of this model is to predict the word that is missing in this window [38]. For example, in the sentence 'the quick brown fox jumps over the lazy dog', a CBOW model that has the goal of predicting 'fox' with a window of 2 would receive 'quick brown jumps over' as input. In a Continuous Skip-Gram architecture, the reverse is attempted: one tries to predict the window of words around a word to learn a word representation [38]. A visualisation of these architectures by Mikolov [38, 39] is shown in figure 5. These word representations allow for an embedding of words that contains some contexts in which a word is used, and allows for words with similar meaning to gain a similar embedding through a Word2Vec model. Using this hidden layer, the word is transformed into a vector of the length of the hidden layer. For documents, this creates a vector for each word in the document that is in the vocabulary of words on which the Word2Vec model was trained. To create a representation of a document using Word2Vec, the element-wise mean of all elements within a document is used [40]

FastText
A variation of Word2Vec also incorporates character-level n-grams, which learns character embeddings as well as word embeddings, which decreases the impact of spelling errors, affixes and suffixes [41]. This variation, *FastText*, creates an embedding of a word based on the sum of the embeddings of character-level n-grams of words. An n-gram is a consecutive sequence of N tokens or in this case, characters. For example, the word-level 2-grams or 'bi-grams' in 'the quick brown fox' would be 'the-quick', 'quick-brown', 'brown-fox'. The same transformation method as Word2Vec is used to transform a document with word embeddings into a single input vector, through the element-wise mean of the word embeddings.

Word2Vec + TF-IDF & FastText + TF-IDF The word embedding methods and TF-IDF weights can be combined into one transformation method, where first, each word embedding is multiplied by the equivalent TF-IDF weight for the word, after which the elementwise mean is taken of weighted word embeddings vector to create a vector representation of the document. This allows for a focus on word embeddings important in the classification of a document. This method has been described for the combination of FastText and TF-IDF weights in the context of a Topic Aware Mixture of Experts (TAMoE) approach to improve the focus on important elements large text documents extracted from Wikipedia and Wiki-how to improve captioning of videos [42]. Therefore, it provides a promise of improving on standalone word embeddings or TF-IDF weighting.

Doc2Vec
Whilst Word2Vec and FastText allow for the embedding of words which capture some higher-level features of the text which TF and TF-IDF are unable to capture, all of the methods above lack the ability to capture the context of words into its numerical encoding. *Doc2Vec* implements similar concepts as Word2Vec and FastText, as it uses the window around words to create a hidden layer from which an embedding of a word can be derived. To provide some sense of context to the word embedding, Doc2Vec models also add a paragraph ID to the window of words around a word, where the paragraph ID is often a label for a document. These paragraph IDs provide additional information about the topic in which a word is currently residing, and can improve the embedding. These *Paragraph Vectors* are shown to be a strong contender to vectorization techniques such as FastText and Word2Vec as well as weighting techniques such as TFIDF [43].

BERT & ELMo
The most state of the art models for word embeddings are *Embeddings from Language Models (ELMo)* and the *Bidirectional Encoder Representation from Transformers (BERT)*. These differ significantly from TF, TF-IDF, FastText and Word2Vec, as ELMo and BERT utilise the entire context around a word when determining the embedding vector, whereas other methods can infer a word vector from a single word [44]. For example, the word 'cell' in the sentences 'The prisoner in the cell was unhappy', 'Mitochondria are the powerhouse of the cell' and 'I can put numbers in an excel cell automatically with RPA', would have the same vectors in a Word2Vec model, but different vectors in BERT and ELMo embeddings.

Where FastText and Word2Vec create one vector for a word, regardless of context, the vector of a word can differ when used in a different sentence in BERT and ELMo. The largest downside of the BERT and ELMo language models is the required amount of time and corpus size to reap the benefits of these more complex language models.

3.2.2 Classification

This study utilises four different models for classifying text. These models all attempt to create a decision boundary around textual features in a different way. In this section, these four models are discussed.

Support Vector Machines
Scalar Vector Machines (SVMs) were made to

learn an optimal separation between two separable classes using input vectors which are non-linearly mapped to a high dimensional feature space [45]. In a binary classification problem, an SVM would create a decision boundary, or hyperplane, with the greatest margin between these two classes. This margin is defined as the sum of the distances to the hyperplane from the closet points of the two classes in this binary classification [45]. The data points are referred to as *support vectors* [45].

If the binary classification problem does not involve linearly separable classes, the SVM tries to find the hyperplane that maximises the margin whilst minimising the total number of misclassification errors. This trade-off is controlled by a parameter C, where $C > 0$ and the error tolerance of the SVM becomes higher with lower values of C [45]. Choosing an appropriate value for C allows for better generalization to reality by minimizing overfitting. Choosing a high value for C might result in a better fit on the training dataset, but not to data outside of the training dataset.

Moreover, the parameter γ is used by SVMs, where γ determines how many support vectors should be taken into account to fit an optimal hyperplane on [45]. A γ which is too large will result in taking into account too many support vectors such that no regularization using parameter C can occur, resulting in guaranteed overfitting. A γ which is too small will not allow an SVM to capture any complexity from the available support vectors. It would result in a small number of support vectors determining the separation from a large number of training examples.

SVMs, as well as other classifiers, can use input data from a binary classification problem and non-linearly map these to a high-dimensional feature space that creates a linear classification problem in that feature space. Through these, a non-linear decision boundary can be made using linear separation in a set of high-dimensional features [46]. After transforming, the data is not represented individually, but through a set of pairwise comparisons [47]. Kernel functions are used to reduce the computational costs of these non-linear mappings to high dimensional feature vectors of input vectors [48].

Kernel functions available in sci-kit learn, a popular python package to implement SVMs, are the linear, polynomial, radial-based and sigmoid kernels, based on different ways of pairwise comparison [49]. The polynomial kernel function requires an additional parameter d describing the number of degrees of separation, where $d = 2$ usually forms the best input parameter for NLP tasks, as higher values tend to cause overfitting [50]. Moreover, the polynomial and sigmoid functions allow for the implementation of bias, r, which determines the trade-off between higher-order kernels and lower order kernels.

SVMs are deployed for binary classification problems, involving only two-classes. For multi-class classification problems, as is the case in this study, different techniques can be deployed. One-versus-Rest (OVR) classification separates the multi-class classification problem into a set of binary classification problems by taking each class and training a classification estimator, such as an SVM, against the rest of the classes [51]. One-versus-One (OVO) classification, $N(N-1)/2$ classifiers are trained for a multi-classification problem with N-classes, where each class is trained against another class, and based on +1 voting of each classifier, a classification can be made on an unknown sample [51].

Random Forest

A random forest is an ensemble of tree classifiers where each of these tree classifiers is generated using a random independent sample from the input vector, and a combined vote of these tree classifiers determines the class classified by the random forest [52]. During the generation of trees, a random feature is selected or a combination of features at each node in the tree to grow the tree further. Random Forest is suggested to always have a converging generalisation error with a growing amount of trees [53]. Creating a tree in a random forest requires a criterion on which features are selected and how a tree is pruned. Popular criterions are the Gini-index and Information Gain Ratio.

Both calculate an impurity of a feature to evaluate if it belongs to a certain class. Gini

index is used by [52] due to its simplicity, however, Gain ratio can prevent excessive bias towards small splits, as the gain is normalized with the attributes entropy [54].

Several other parameters also have to be user-defined, such as the amount of trees, the depth of the trees, the amount of features required to split a node, the maximum amount of features that should be considered per node and the minimum amount of samples that can create a leaf node, the end of a tree. Moreover, one can choose to use a sampling technique called bootstrap replication, where only a subsample of the training data is used to grow a tree [54]. Finding an optimal set of these *hyperparameters* can be done using a grid-search, where options are given for each hyperparameter and all possible combinations are checked. A less thorough but often quite as effective technique is using a random grid-search, were given a limit of options to be checked, only part of all possible combinations is checked.

Gradient Boosting Machine

Gradient Boosting Machines (GBMs) are greedy function approximators, where for each tree, gradients are assigned to each node from which a loss is calculated describing the deviation of the prediction to a true value [55, 56]. Moreover, trees are grown greedily to improve on previous ensembles of trees and grown in an additive manner, as opposed to Random Forests, which grow at random and independently. In general, GBMs tend to have high bias and low variance in trees, whilst random forests have low bias and high variance in trees. In general, trees in a random forest tend to overfit, but increasing the number of trees tend to prevent this [53]. In GBMs however, increasing the number of trees can result in overfitting, as the variance is not an issue, but the bias when selecting trees might become an issue. In this study, GBMs are implemented through the eXtreme Gradient Boosting system 'XGBoost', an implementation which allows for faster computation optimized for parallel computation and has been used in a wide variety of prediction challenges and applications [56]. This study will not refer to GBMs, but the implementation XGBoost in further sections.

Bilinear-LSTM

In all previously mentioned classification models, features are samples independent from each other from an input vector. For natural language learning, this is referred to as 'bag-of-words', where a document of text is transformed into an unordered 'bag' of features. A neural network architecture that utilises a Bilinear Long short-term memory layer (Bi-LSTM) uses a different approach, where a classification boundary is learned by analyzing features in the context of each other. An LSTM is a solution to the vanishing gradient problem suffered by Recurrent Neural Networks (RNNs). LSTMs preserve a memory of previous inputs, preventing the explosion or vanishing of gradients [31]. Moreover, LSTMs have proven successful in text classification tasks [57].

RNNs can map historical inputs to outputs using a historical hidden layer, whereas a standard perceptron can only map an input to output vectors. This allows a neural net with enough RNNs to learn to generate sequences, as these recurrent connections allow previous inputs to be preserved within a neural network through its hidden layers[58]. Moreover, Bidirectional RNNs (BRNNs) generate two sets of hidden layers from an input, representing both forward states and backward states of the input [59]. In the context of Natural Language Processing, this allows a BRNN to utilise words prior and after an input word. Whilst it would seem that long-range dependencies in input data can serve an issue to LSTMs, as only part of the historical information is preserved, [31] has shown that properly preprocessing data such that contextual long-range dependencies are captured in short-range dependencies only slightly improves accuracy in a learning task [57].

3.2.3 Imbalance

Class distribution is an important element in several text classification problems, for example in the classification of spam e-mails [60]. In a binary classification problem such as spam detection, the class imbalance can be described as a

ratio against a majority negative class, in this case, normal e-mails. The imbalance ratio would be 1 spam e-mail against 10 normal e-mails, described as 1:10. The imbalanced distribution of classes is not the only data characteristic relevant in imbalanced datasets [61, 62]. The effects of class imbalance diminish if enough data is available [62]. Class overlap, which occurs when two or more classes within a classification problem are very similar, can decrease the amount of the minority class being classified correctly [63]. Linearly separable classes are suggested to not suffer from class imbalance at all [62]. Finally, within-class concepts can further increase the effects of class imbalance. Here, within-class concepts are underlying subconcepts of which one class is compromised. For example, if a classification between different animals has to be made, each breed of dog would represent a sub-class of dog. The presence of such sub-classes increases the complexity of the classification problem and has a reinforcing effect on the class imbalance problem [61, 64].

Neural Networks, Support Vector Machines and Decision tree algorithms such as Random Forest and Gradient Boosting Machines all suffer from class imbalance. However, Support Vector Machines have been reported to suffer less from imbalance problems compared to other algorithms, since only a few support vectors are used to determine decision boundaries [62]. Several methods exist to prevent class imbalance. In this thesis, only the cost-sensitive approach of weighting samples based on class imbalance has been used due to limited computational capacity. However, to give directions to potential improvements to be made to the model, several sampling methods are also introduced. Finally, it has to be disclaimed that boosting approaches and cost-sensitive boosting approaches by changing the optimisation method of various machine learning models that are based on loss or ensemble can be used to reduce the effects of class imbalance. These are not further introduced in this thesis but could provide a future direction for improvement of the final solution.

Weighting

Weighting is a method where each sample receives an additional weight depending on the class distribution, where the weight is defined by equation 2.

$$w_i = \frac{S}{N \times s_i} \tag{2}$$

Here, S is the total amount of samples in the dataset, N is the number of classes and s_i is the amount of samples of a specific class i, where $s_i \in S$. Weighting each sample increases the cost of misclassification of a sample of class i by w_i.

Undersampling

Undersampling is a method where instead of choosing an algorithmic solution, the data is manipulated to prevent the class imbalance problem by removing part of the majority class and reducing the imbalance ratio [65]. Several techniques exist, such as *random undersampling*, where random samples are taken away from the training data, or *distance-based undersampling*, where samples of the majority class closest to the minority class are kept [66]. Undersampling is often applied when datasets are already large, and oversampling would only further increase training time. Undersampling can then be used to both reduce training time, as well as improving performance [66].

Oversampling

Oversampling is the opposite of undersampling, as additional samples are generated for the minority classes. As with undersampling, various techniques exist to generate additional samples. *SMOTE* is a method for oversampling where the minority class is over-samples by creating synthetic examples along line segments joining any or all of the k minority class nearest neighbour [67]. A limitation of SMOTE, however, is the low variance in the created synthetic samples that would otherwise be present with new samples for the minority dataset, in reality, creating an overfitting bias. *ADASYN* improves on this by creating more examples of a minority class of samples that are specifically difficult for classification algorithms to classify [68]. This improves

on SMOTE by preventing bias and putting the focus of the learning task on difficult examples. A large downside of oversampling however is the increase in required training time and computing power as the total size of the dataset increases.

3.2.4 Evaluation metrics

To evaluate the experiments, an evaluation criterion is required. The following scores, metrics, as well as the validation methods, are commonly used, valid metrics and methods to assess the performance of machine learning models [69]. A commonly used metric to evaluate machine learning models is the F1 score. The F1 score is the harmonic mean between precision and recall. Precision and recall are calculated using True Positive (TP) where a prediction for a positive class is done correctly, True Negative (TN) where a prediction for a positive class is done correct, False Positive (FP) where a positive class is misclassified as negative and False Negative, where a negative is misclassified as positive. An overview of these for each of these can be made using a confusion matrix, presented in table 2 The predictions in the confusion metrics are done using a holdout part of the data used for validation. This means that only part of the data was used for training.

Table 2: *Example confusion matrix.*

		Predicted	
		N	P
Actual	N	TN	FP
	P	FN	TP

Using the variables in the confusion matrix, precision can be calculated using equation 3, and recall with equation 4.

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$recall = \frac{TP}{TP + FN} \tag{4}$$

Here, high precision means that a low percentage of positive classifications belong to the negative class, and high recall means that a low percentage of the positive class is classified as the negative label.

From these two metrics, the impact on the functionality of the final solution can be derived. With low recall, the solution will have little impact, as only a low amount of question classes are predicted when they should have been. With low precision, a lot of predictions of the question classes are done in error, resulting in the wrong action being sent to the employee.

In multi classification problems, one has to choose to average the scores of each label on either a micro or macro level. The F1 score is used to illustrate the difference between the two approaches. Precision and recall can be calculated in the same way for multiclass problems. By calculating the score on a macro level, one calculates the F1 score for each class and takes the mean to calculate the macro F1, given by equation 5, where i is the i-th class and n the total amount of classes.

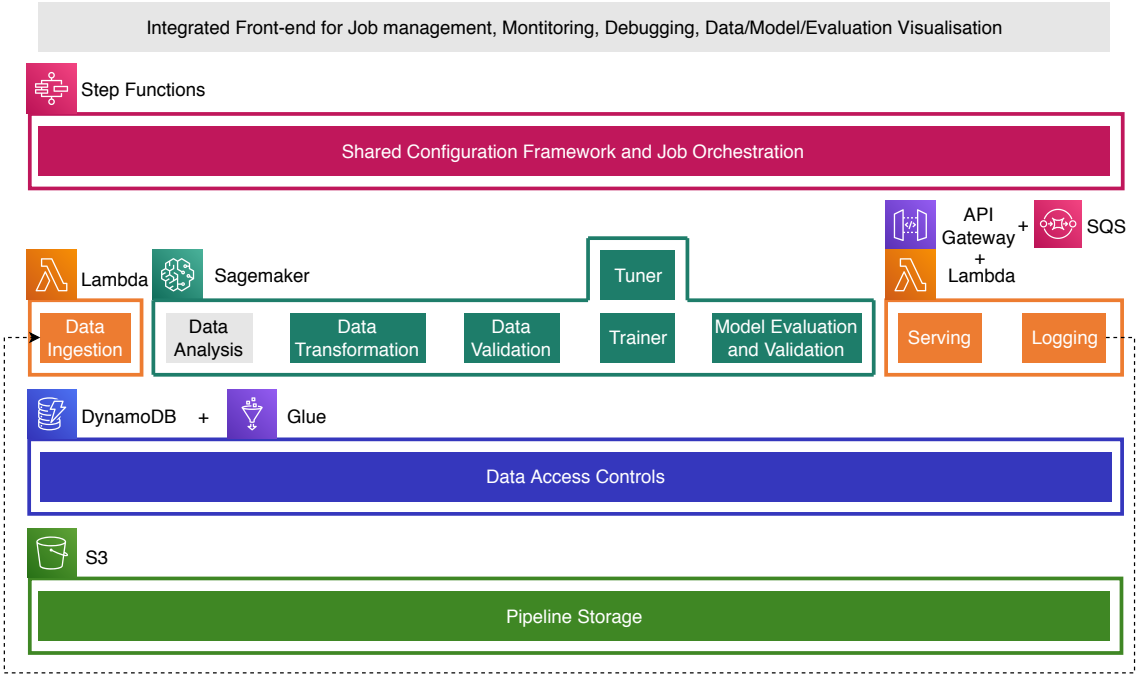
$$macroF1 = \frac{1}{n}\sum_{i=1}^{n} F_{0.5}(precision_i, recall_i) \tag{5}$$

By calculating the score on a micro level, one combines all the True Positives, False Negatives and False Positives of each class. Using these combined True Positives, False Negatives and False Positives, precision, recall and subsequently an F1 score are calculated.

To validate the results of a model with imbalance, shuffled stratified k-fold cross-validation can be used. In k-fold cross-validation, the dataset is split into k training sets and test sets, where the test sets are independent of each other. This means that for a value of $k = 5$, the dataset is split into 5 different test sets, each independent and of $\frac{1}{5}$th the size of the total dataset.

Stratifying these k-folds ensures that of each class, the imbalance ratios for each class in the full dataset are preserved across the training and test sets. Besides, the data is shuffled. This ensures that the order of the data does not influence the validation score.

Figure 6: *The high-level component overview of a machine learning platform as defined in the TFX paper [70] and Amazon Web Service functionalities to support these components.*



3.3 Operational Machine Learning and Serving

In this section, concepts from the TensorFlow Extended (TFX) Platform is used to uncover the challenges related to operational machine learning in production and ways to overcome these challenges. TensorFlow Extended is a general-purpose machine learning platform implemented by Google that supports machine learning models for a wide variety of tasks [70].

The TFX platform was built with several challenges in mind, some relevant to the question answering system developed in this study. *Continuous training and serving* involve the challenges that a machine learning model is not only deployed and used once but needs to be updated to remain up-to-date whilst actively serving end-users. *Human-in-the-loop* the system needs to be easy to evaluate by functional maintenance users who evaluate the day to day performance of the model, but also by machine learning experts who can improve the existing model. *Production-level reliability and scalability* involves resilience to disruptions and inconsistent data, software, user configurations and failures and moreover, the platform needs to be able to scale-up and down depending on end-user demand. [70]

To combat these challenges, TFX was designed with the high-level component overview illustrated in figure 6 in mind. Compared to the original TFX template, garbage control has been left out, as this is dependant on the cloud platform on which the TFX template runs. The TFX components serve as a template for the high-level solution design of the question answering system in this thesis. In the rest of this section, these components are described. Moreover, the implications for the high-level solution design are discussed, as well as services on the Amazon Web Services (AWS) platform that can aid in facilitating the functionality of these components. The impact of each of these services on the high-level components of a machine learning platform is vi-

sualised in 6.

3.3.1 Amazon Web Service Functionalities

AWS Lambdas can be seen as pieces of code which are executed only when triggered. AWS Lambdas can be used for various purposes, one popular being data processing. A lambda can be triggered by a web end-point sending data to the lambda, after which the lambda takes care of cleaning and transforming the data before storing the data or processing it further [71]. These lambdas can serve as the pipelines that connect the components of the machine learning platform with each other and the outside world during data ingestion and serving.

AWS API gateway allows for the creation of secure API end-points, for example for Amazon-based web applications or AWS Lambdas. These allow AWS Lambdas to be exposed to the web, with the necessary security in mind [72].

AWS Simple Queue Service (SQS) is a messaging queue service that allows scalable cloud applications to deal with the overhead of message-based applications. This can send, store and receive messages from any software component at any volume, without losing messages or requiring other services to be available [73]. With SQS, incoming requests from the API gateway can, through a lambda, create a queue for tickets to be inferenced, and allows tickets only to be removed from the queue if the response has safely arrived at any other software component, allowing for reliability when processing data from microservices.

AWS Simple Storage Service (S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance [74]. As lambdas do not store objects between triggers, S3 can be used to store data, models and possibly other objects that need to persist throughout the operation. S3 is secure, GDPR compliant, easy to scale up and down and reliable in service, making it the go-to choice for data storage in cloud applications made with AWS [74]. As such S3 is used as the service responsible for the pipeline storage for the training dataset and the serving model.

Amazon DynamoDB is a solution that offers storage and real-time manipulation of large databases. Here, the master training data is stored and necessary changes to the database are done, such as updating labels or adding new training samples [75].

AWS Glue is the connector between the DynamoDB where the master data is stored and AWS S3 where the training data is stored on which the model can train and validate itself. AWS Glue allows for extraction, transformation and load (ETL) pipelines conventionally used to prepare data for analytics in practice [76].

Amazon SageMaker is the machine learning platform of Amazon, which supports a wide variety of machine learning platform components out-of-the-box. As with every AWS functionality, it is scalable on-demand and allows access to computing power that enables fine-tuning highly complex models which would take a considerable amount of time to tune otherwise [77]. Moreover, Amazon SageMaker allows for the deployment of trained machine learning models in production, and using AWS lambdas, can make inferences on live data and return the results.

Amazon Step Functions helps developers build, run, and scale background jobs that have parallel or sequential steps [78]. This service can be used for job orchestration of the retraining of serving models using a shared configuration framework.

Whilst an integrated front-end can be created for this solution, this element was regarded as out of scope for the design of the system. Therefore, no AWS services have been identified that can do this part of the TFX framework.

3.3.2 High-Level Machine Learning Platform Components

Data ingestion refers to the method through which data is acquired by the machine learning pipeline [70].

Data Analysis involves the extraction of descriptive statistic from the ingested data. This allows for tracking of changes in the data over time and can aid debugging of the data [70].. This was

done during the thesis, but not implemented as part of the automated machine learning pipeline of the solution design.

Data Transformation involves the transformation of the extraction of features from data [70]. Methods to do this have been described in section 3.2.1 'Transformation'.

Data Validation involves the tracking of anomalies in data, for example, unknown labels or unknown features. Through this, one can prevent potential errors in the training workflow [70].

Training & Tuning involves the training of a model on the training data and tuning the model hyperparameters depending on the performance whilst training. Training a model can be time-consuming, especially when there is a lot of data to train on. A technique that can prevent the model from having to be retrained from the ground up is *warm-starting* [70]. Warm-starting uses parameters from the old model to initialize a new model and uses new training data to train the new model, preventing having to retrain on all data [79]. Warm-starting is not always available for machine learning models, and for deep learning models, effectiveness depends on the generalisability of the old context to the new context of the model [70, 80].

Model Evaluation and Validation involves evaluating the trained model against hold-out test data [70]. Whilst the serving model will be trained on the full dataset, evaluation requires training on only part of the data. The validation and evaluation techniques used in this thesis are discussed in section 6 'Experimental Setup'. Based on the evaluation results, one decides if the existing model can be replaced by the new model or not. Moreover, one can select the best transformation method and classification model for the current training data based on a common evaluation score.

Serving involves exposing the model to new input and serving an inference from the serving model [70].

Logging involves recording the decisions made by the serving model and using these for future training and optimization of the model.

Data Access Control describes how data can be extracted from the platform to both serve a training dataset or allow for further analytics.

Pipeline Storage holds the current serving model, training data and other data objects necessary to serve and train in production.

Shared Configuration Framework and Job Orchestration defines what the configuration of the machine learning pipeline is and what type of jobs should be done to make the pipeline work. For example, a configuration for text classification will deviate from topic modelling or clustering. Job Orchestration involves splitting the machine learning tasks into jobs and allows for the machine learning pipeline to be executed in parallel and scalable in production.

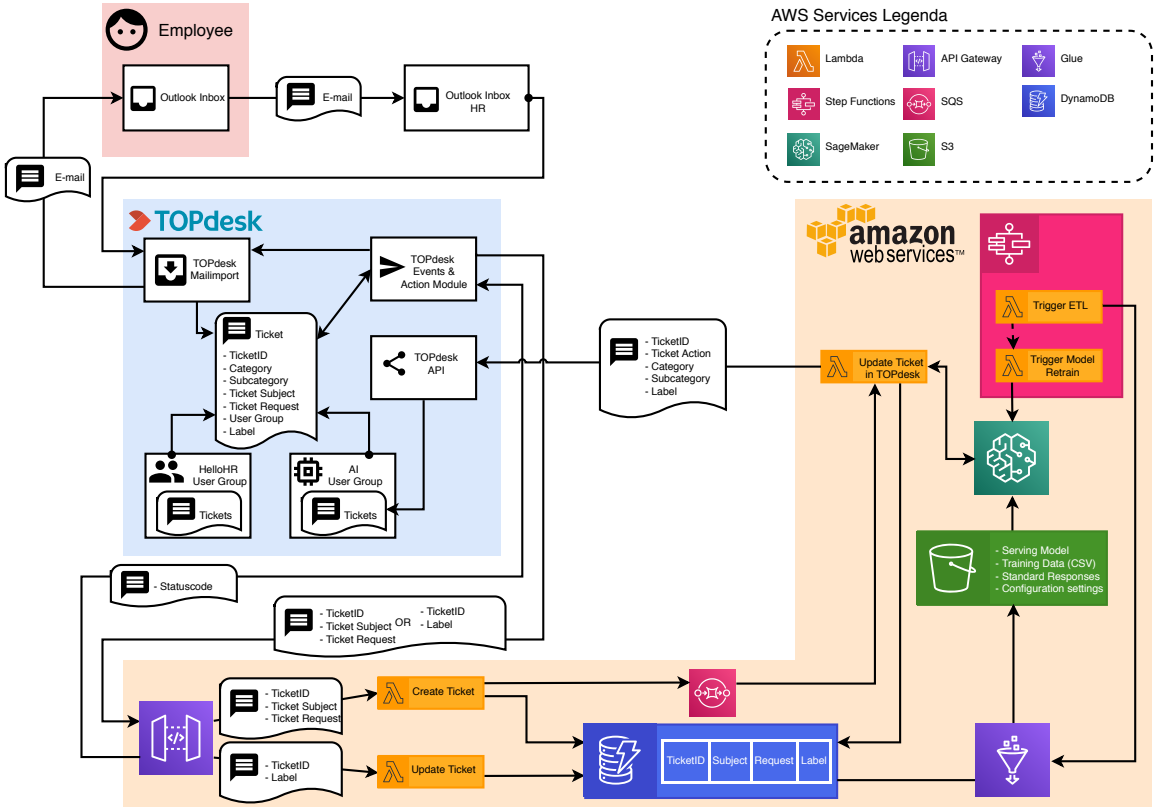
4 Solution Design

In this section, the full solution design for the client is described. All flows and logic behind the solution are discussed, giving insight into what is required to operationalise a question answering system at the client. An overview of the solution design can be found in figure 7. The full solution design reveals the full context in which the QA system will exist after implementation. The configuration has been evaluated and co-created with various AWS cloud architects from VodafoneZiggo. Moreover, with the help of two consultants from TOPdesk, an API has been designed and implemented with the help of aforementioned AWS cloud architects that allows for the full communication between the two cloud solutions as visualised in figure 7 and described below. Except for the Step Functions, SageMaker Module and Glue services, the entire architecture has been tested and validated. A demo and specification of the design can be found in Appendix A.

4.1 Employee Questions: From Employee to TOPdesk

Whenever an employee has a question, an email is sent to the e-mail address of HelloHR. These employee emails are acquired by the TOPdesk Mailimport module, which transforms these e-

Figure 7: *Solution Design of the Question Answering System in operation at the client, using an AWS Cloud Architecture.*



mails into a ticket and imports these into the TOPdesk environment. Here, a ticket is assigned to a user group on the event of an e-mail import through the TOPdesk Events & Action module. On import, one of two events can occur, 1) the ticket is a new ticket, which is assigned to the AI user group[1] or 2) the ticket is a response to an answered ticket[2] by the AI, after which the ticket is moved from the AI user group to the HelloHR user group, where it can be answered by an HR employee.

In case 1, a message will be sent containing the ticketID, the ticket subject and the ticket request to the AWS API gateway. This will trigger the 'Create Ticket' event.

In case 2, the HR employee is notified of a potential mistake made by the Question Answering System through a message hidden in the TOPdesk ticket, which will not be used in the response towards an employee. This message indicates to check the label assigned to the ticket by the QA system, and if necessary, change it to a correct labelling. When a label is corrected, a message is sent to the AWS API gateway containing the ticketID and the changed label, triggering the 'Update Ticket' event.

[1]This is a user group made for the Question Answering System
[2]Responses on the same ticket are recognized by a ticketID in the subject of an e-mail

4.2 Create Ticket: From TOPdesk to AWS

Accessing the API Gateway of the AWS web service for the QA system with a request involving fields for the ticketID, ticket subject and ticket request will trigger an AWS Lambda that will queue the ticket using SQS and store the ticket in a table in DynamoDB with an empty label field. The queue will trigger the Lambda 'Update Ticket in TOPdesk' which will ask for inference from the SageMaker module. The SageMaker module will acquire the serving model and infer a question class from the ticket subject and ticket request. The classification will be stored in DynamoDB by the same Lambda. Depending on the classification, one of two things will happen, 1) the serving model was able to infer a question class from the ticket subject and ticket request or 2) the serving model was unable to infer a question class.

In case 1, a request is sent to the TOPdesk API which changes the ticket action, category, subcategory and label using the ticketID. The action, category, subcategory and label values all depend on the inference of the serving model. A ticket can only be changed this way when the ticket is in the AI user group; if the ticket is moved during inference, nothing will happen.

In case 2, a request is sent to the TOPdesk API which changes the user group to HelloHR and sets the label to 'Out of Scope', meaning that the serving model was unable to classify the ticket. How this event can occur will be revealed in the 6 'Experimental Setup' section, in which a PoC version of the serving model will be developed. In addition to changing the label and user group, a hidden message will be left asking the HR employee to check the label and if necessary, correct it by assigning the right question class label. This will trigger the 'Update Ticket' event.

4.3 Answering Questions: From TOPdesk to Employee

When a response has been received by AWS, either one of two things happened: the ticket has been updated by AWS within the AI user group, or the ticket has moved from the AI user group to the HelloHR user group, where it eventually will be answered by an HR employee. Regardless, a response will eventually be sent by e-mail to the employee through an existing event trigger that occurs whenever an action is added to a ticket. This e-mail contains the question of the employee, the action of the HR employee or QA system, and the ticket subject extended with the ticketID.

4.4 Update Ticket: From TOPdesk to AWS

Accessing the API gateway of the AWS web service for the QA system with a request involving fields for the ticketID and label will trigger an AWS Lambda that will update the label in the DynamoDB table. This will allow HelloHR employees to correct misclassifications and improve the quality of the training data. Moreover, as labels can be added in TOPdesk, new labels can be identified on which the model can be trained, allowing the serving model to grow in scope over time.

4.5 Improving the Serving Model

Over time, it is expected that the serving machine learning model will deteriorate in quality. As such, scheduled step functions can be executed that will trigger the Lambdas 'Trigger ETL', which triggers a Glue ETL from DynamoDB to S3 creating a fresh training dataset using the ETL parameters specified in Glue. After this ETL, the Lambda 'Trigger Model Retrain' will be triggered, which will, in turn, trigger the training towards a new serving model.

Improving the serving model heavily depends on the quality of labels and the number of new labels assigned by HelloHR employees through the 'Update Ticket' procedure. To help HelloHR employees get a feel for labelling and make them aware of the changes to their workflow as a result of an implementation of the solution design, a training has been made that addresses the topics of changes to TOPdesk, how to label and how to improve the serving model, referred to as 'HelloAI' internally.

Here, the changes in the workflow are described which involve the hidden message and new labelling option within the ticketing user interface. Moreover, the definitions of various question classes are elaborated, as well as the ability for HelloHR employees to request changes to the standard responses based on a question class by HelloAI or and the ability to add new labels to the system.

To facilitate this, a functional maintainer will be assigned that can assess the new labels and standard responses by HelloHR and adjust these accordingly in the system.

5 Proof of Concept

Aside from developing the solution design, this study investigated a Proof of Concept for the QA system through statistical difference-making experiments [23] involving a variety of machine learning models that form the core of the question answering system. The approach to these experiments in the context of the client will be discussed in section 6 'Experimental Setup', and the results in section 7 'Results'.

In this section, the progress towards the first experiment is described. First, the nature of the data used for the experiments is discussed, as well as the steps taken to clean the data to acquire a target dataset. Second, a method is described through which labels were discovered and assigned to the target dataset using a mix of manual work and machine learning. Finally, the classification problem is described, as well as a Proof of Concept which can function within the full solution design, albeit with a limited scope.

5.1 Target Dataset

A description of the acquired dataset can be found in Appendix B 'Initial Dataset'. In total, there are 91464 entries in the dataset. Whilst there are category and subcategory labels within the dataset, these do not correspond directly to a standard answer. Therefore, labels for specific question classes still need to be generated. Before the derivation of labels from the dataset is described, the cleaning methodology used to create a target dataset is briefly described. For a more extensive description of the cleaning of the request and action columns, including regular expressions, code and other material used, consult Appendix C 'Details Text Cleaning'

5.1.1 Cleaning

The scope of the target dataset is all tickets *received by e-mail* , that have a *Request* and *Action*. Based on the following filters, a target dataset was created. First, all rows with the value 'False' in the field 'Done' were dropped. Second, all rows with a value 'NaN' in either 'Request' or 'Action' were dropped. Finally, all rows where the 'Request' field did not contain any e-mail headers were dropped.

After filtering, we are left with a target dataset of size 77000. Several issues remain in the data. First, the fields request and action are cut off at around 4000 characters. For the request, this does not present an issue, but for the actions it does. When a person sends an e-mail to HelloHR, their action is only registered once. When new e-mails are sent for the same ticket, these requests are added to the top of the action field. This means that the original action can fall beyond the cutoff and become irretrievable. Whilst this problem cannot be overcome, one thing is for certain. The action to the original request can be found at the last TOPdesk header in the action column.

Second, now that the original request and action are retrieved, several cleaning steps follow to remove noise from the text. An example of noisy text can be found in example 1. First, handles are placed to indicate what should be removed

Example 1: *A request made on January 18th of 2018 as it appears in the dataset.*

```
dd - mm - yyyy hh:mm <lname>, <fname> <lname prefix>:
Datum verzonden: dd - mmm - yyyy hh:mm
Naar: "HelloHR@<company>.com" <HelloHR@<company>.com>
Onderwerp: YOUFORCE inloggen

Hoi collega's,

Het lukt mij niet om in te loggen, krijg steeds de melding dat mijn e-mail adres niet
te herkennen is.

Kunnen jullie mij een link sturen om een nieuw account aan te maken?

Groeten,
<name>

<contact information>
```

Example 2: *Handles are placed in order to remove clutter.*

```
<HEADER>

<OPENER>

Het lukt mij niet om in te loggen, krijg steeds de melding dat mijn e-mail adres niet
te herkennen is.

Kunnen jullie mij een link sturen om een nieuw account aan te maken?

<SIGNATURE>
<NAME>

<contact information>
```

Example 3: *Using regular expressions, the core message is left over.*

```
YOUFORCE inloggen

Het lukt mij niet om in te loggen, krijg steeds de melding dat mijn e-mail adres niet
te herkennen is.

Kunnen jullie mij een link sturen om een nieuw account aan te maken?
```

Example 4: *After a final cleaning step, most noise has been removed*

```
youforce inloggen lukt loggen krijg melding e mail adres herkennen link sturen nieuw account
```

in the text, such as headers, openers, signatures and names. Within example 1, all information is anonymised by handles. Handles placed and used for clean are capitalised in example 2. After placing these handles, regular expressions are used to remove some noise from the text. At the end of this step, the subject of the ticket is added to the top of the message[3]. The resulting message after this step can be seen in example 3.

Third, additional variants of clean text are made, involving different n-grams. N-grams have been introduced in section 3.2.1 under 'FastText',

[3]names are also removed from the subject

and are collocations of words that occur together a significant amount of time compared to separate occurrences. Three levels of n-grams are created: unigrams, where no words are collocated, bigrams, where 2 words that are significantly collocated are combined and trigrams, where on the level of 2 and 3 words combinations are made if these words are significantly collocated.

Finally, the text is tokenized using SpaCy [81], an NLP package for python supporting the Dutch language, which also extracts the Part of Speech (PoS) and lemmas from individual words in a text. A lemma is the dictionary form of the word and is used to normalize various instances of words into one term, for example by making plural or past tense similar to the singular and present tense words. Using the tokenized words, we preserve only the lowercased lemma of a token if the token is either a verb, noun or adjective and if the token is not a stopword and if the token is an alphabetical character, not a form of punctuation or number. Example 4 shows the final result of cleaning the text. The cleaning might seem too rigorous, as words such as 'not' (niet) were removed from the text. Arguably, this has no significant impact on the classification task, assuming everything arriving is by default an issue. However, in cases where multiple questions were being asked, indicating for example that 'I could log into this system but not into that system', important information is lost. Whilst this will not matter in the bag-of-words models, the sequence-based models such as the Bi-LSTM Neural Net might perform sub-optimally due to this cleaning behaviour.

5.1.2 Labelling

As the initial dataset does not include any labels, labels still had to be created to make the supervised learning-based models introduced in 3.2.2 'Classification' work. Based on the initial e-mail automation proposal by the consulting firm where a proof of concept would involve classification across seven categories, the goal was set to retrieve at a minimum of seven labels. In this section, the labelling strategy is elaborated, involving label discovery, label expansion and label quality.

Label Discovery

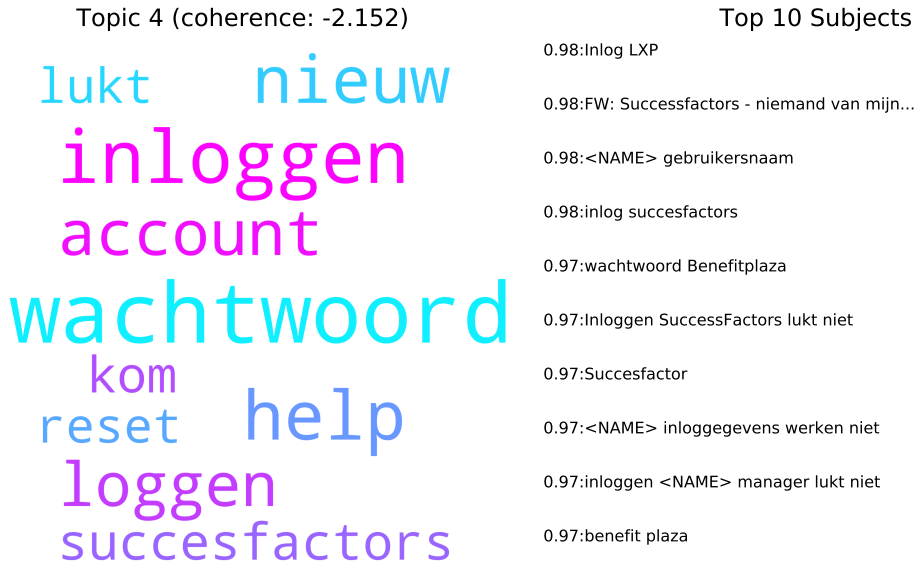
For label discovery, keyword extraction and search was applied to the action part of the ticket. First, however, Topic Modelling using Latent Dirichlet Allocation (LDA) was applied to explore the contents of the dataset and find coherent topics across the questions of employees.

An LDA was used as a tool to explore the dataset and find common topics. This was done using Gensim, a Topic Modelling package for Python [82]. An LDA is an algorithm that approximates a Dirichlet distribution for words in a topic k and the Dirichlet distribution of topics in a document d [83]. Details about the LDA configuration can be found in Appendix D 'Hyperparameters'. For exploration, visuals were made such as in figure 8. Here, the subject was printed, together with a word cloud of the topic and the coherence score, indicating the quality of the topic, allowing for the comparison of quality across topics within the topic model. The topic presented in figure 8 is an example of a relatively coherent topic, indicating the relevance of login problems within the employee questions. This concluded a rough exploration of the dataset, as the resulting topics were not reliable enough or exclusive enough to label an initial set of data. Next, several actions were taken to discover and label an initial set of labels in a reliable manner.

First, a list of existing standard responses was retrieved from the HelloHR department. The utilisation of these standard responses, however, was not consistent, causing few results to show up from these responses. In an attempt to improve and automatically label a large subset of questions, the TF-IDF scores for complete sentences were retrieved and inspected for all subcategories. This resulted in initial labels and discovery for the classes *sf_loginissue*, *yf_loginissue*, *bp_loginissue* and *ns_greenwheels*. For all these categories, unique sentence combinations were used in the actions that would only occur in a response to a question specifically for that label.

Second, for each subcategory with at least 100 tickets, a list of most important keywords was extracted from the tokenized text and used to

Figure 8: *Topic 4 in a topic model for 10 topics. The topic seems related to login issues, which would become a majority of the discovered labels.*



search for similar actions. These keywords were evaluated against the found topics during the exploratory topic modelling using an LDA model for 20 topics.

This resulted in the labels *wni_request*, *expiring_leave*, *yf_process_issue*, *change_personal_details*, *adjust_leave*, *lms_loginissue* and *changepoint*. For example, keywords such as 'expiring' (vervallen) occurred a lot in the category 'leave' as well as the term 'retract'. Whilst the sentences could lead to automatic labelling with 100% certainty, these labels still required the measure to determine if a ticket belonged to a label or not. A manual labelling process started where keyword search was performed using the found keywords within a category. This involved the following steps.

1. Look at the raw questions and answers with the keywords within a subcategory to see if the keyword is used in a similar set of questions/answers.

2. Define an initial label and start classifying each question/answer pair found using the keywords for this label in a binary way, either belongs to the label or not.

3. Label at least 30 samples for the label to be considered in scope for the Proof of Concept.

After this, eleven question classes were defined with various numbers of support which were expanded in the next step. Note that more question classes could have been found. However, due to time constraints and the scope of a Proof of Concept, it was chosen to limit the number of question labels to eleven, especially considering the amount of time the following step took.

Label Expansion

After the identification, the found labels were expanded using a mix of manual work and a fast way of limiting the number of potential labels through the use of an SVM. Using the found samples for the eleven labels, for each label, an SVM with a low C value was being trained using a One-versus-One classification strategy, the weighted imbalance strategy and as input first transformed actions, and later questions, using TF-IDF. This low C value allowed for a larger

error margin for the SVM [45]. The One-versus-One classification creates a binary classification problem from a multi-class problem. For this explanation, the label to be expanded is referred to as the *positive* class, whilst the rest of the dataset is regarded as the *negative* class.

The SVM was trained on shuffled data using Stratified K-fold cross-validation over five-folds, meaning that for each fold, a different 20% of data was used for validation and 80% for training, whilst maintaining the ratio between the negative and positive class across these folds in both the training and test set. From the validation, a list of *false positives* can be extracted, which involves a prediction of the positive class in the validation data when the data was labelled as the negative class.

A simple user interface was built that would use the index of the false positive to retrieve the question and action from the target dataset, show these to the researcher and allow for an option to either classify it as the target label or not. In some cases, this choice was extended by allowing other potential labels to be used whenever the initial label of the SVM was rejected. For example, whilst labelling the sf_loginissues, login issues of other systems also emerged. This allowed for relatively fast and reliable labelling of the dataset to classify questions.

Note that only one label was assigned to each ticket; when multiple questions were asked, a 'NaN' label was given, as the request was deemed too complex for the scope of the Proof of Concept.

In sum, the following steps were performed to expand each label using an SVM with a low C value with a range of $[0.01 \leq C \leq 10]$ depending on the amount of true positive labels emerging from these false positives during the manual inspection:

1. Transform the clean action tokens using TF-IDF
2. Train an SVM using stratified 5-fold validation and store the false positives
3. Inspect the false positives for any true positives
4. Label instances that are true positives until the list of false positives is exhausted.
5. Repeat until no true positives emerge from the false-positive list, then raise the C value by a magnitude of 10 and continue.
6. Again, repeat until no true positives emerge from the false-positive list, then switch to clean question tokens using TF-IDF as input.

This resulted in the relatively rapid labelling of $\approx$ 8.9% of the entire dataset. The resulting label volumes can be found in table 3, where a 'NaN' label stands for an unlabeled ticket.

This strategy for label expansion, whilst allowing for relatively high quality of labels compared to assigning labels automatically using keywords and by default being faster than manual labeling without an SVM, takes a large amount of time and effort. In the final solution design, part of this effort is integrated in the regular workload and spread across a large team of people, removing part of the required effort to get more question class labels. Moreover, as new labels are to be added after implementation of the final solution design, label identification will also not be required. For this to work however, a first serving model needs to be developed, which is part of the Proof of Concept design.

Table 3: *Overview of Labels in the target dataset.*

Label	**Count**	**%**
NaN	70146	91.1
sf_loginissue	2065	2.68
yf_loginissue	1673	2.17
bp_loginissue	702	0.91
wni_request	531	0.69
expiring_leave	485	0.63
yf_process_issue	376	0.49
change_personal_details	325	0.42
adjust_leave	255	0.33
lms_loginissue	157	0.20
ns_greenwheels	150	0.19
changepoint	135	0.18

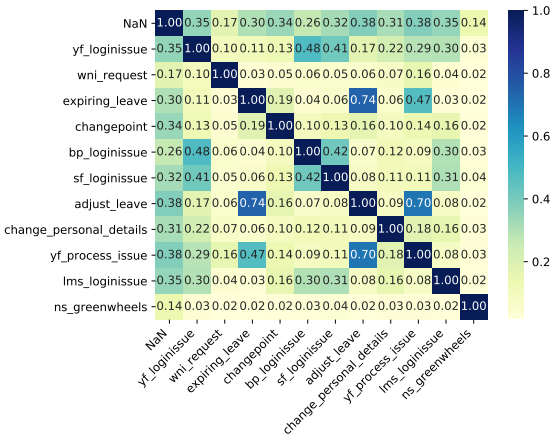
Label Quality
To assess the quality of the labelling, the overlap between labels based on cosine-similarity is calculated, to see if class overlap exists. Class overlap can significantly increase the impact of imbalance in a dataset, causing the minority classes to be misclassified more often [63].

Cosine similarity for two labels is calculated by joining all documents for a label into one document and transforming this using either a Count Vectorization or TF-IDF. Here, TF-IDF is chosen. If $\mathbf{C_1}$ represents a vector with all documents transformed with TF-IDF for class 1 and $\mathbf{C_2}$ represents a vector with all documents transformed with TF-IDF for class 2, the similarity is calculated as in equation 6.

$$similarity = cos(\theta) = \frac{\mathbf{C_1} \cdot \mathbf{C_2}}{\|\mathbf{C_1}\|\|\mathbf{C_2}\|} \quad (6)$$

Here, $[0 \leq similarity \leq 1]$, where two completely similar classes would have a similarity of 1 and two completely dissimilar classes would have a similarity of 0. An overview of the similarity for each class can be found in figure 9.

Figure 9: *Pairwise Cosine Similarity of labels.*



Whilst there is expected overlap with the 'NaN' class, as this contains a wide variety of classes, some other overlap also exists. Between the two labels related to leaving, quite some overlap exists, as well as with the class adjust_leave and yf_process_issue.

Whilst the overlap between two leave categories is obvious, the latter overlap seems strange, but is explainable by the nature of the labels. In some cases, people do not know how to solve an issue that could be solved by adjusting their leave days, or in some cases, people simply do not know that they can adjust leave themselves. In case of the overlap with yf_process_issue, one might want to change leave, but is unable to not because of lacking knowledge, but because the system is not working. This could explain the high similarity between the classes.

Another overlap exists between the log in issue categories, which is also expected. The largest deviation between these categories will be system-specific features or names.

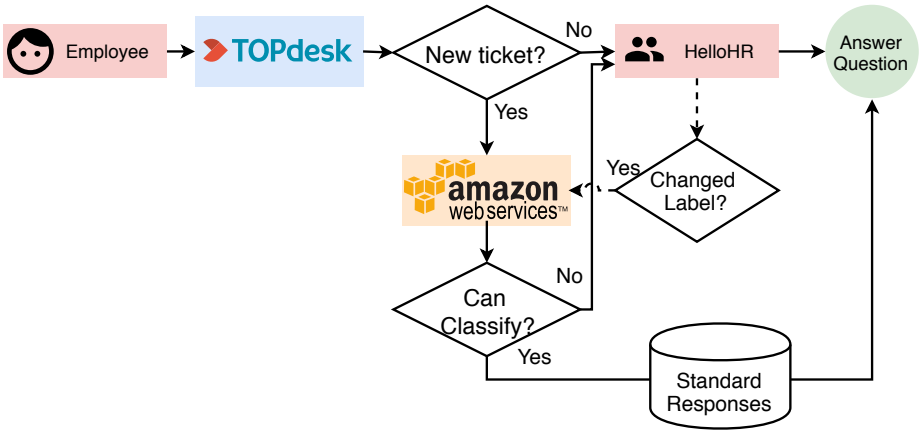
Overall, with some exception, the separation between classes is in a good state, and the exceptions are explainable and not disturbing further progression. Based on the identified labels, a preliminary design of the Proof of Concept can be made.

5.2 Proof of Concept design

Using the twelve labels, a PoC design can be made which can help in investigating both the feasibility as well as the potential of the solution design described in section 4 'Solution Design'. This is visualised in figure 11. First however, these components are introduced using the flowchart in figure 10. Here, the TOPdesk system is able to recognize if a ticket is new or not. If the QA system in AWS can classify the question in the ticket, so a classification of any label except 'NaN', a standard response is retrieved based on the classification and sent to the employee. If the system cannot answer the question, an HelloHR employee can answer the ticket. Moreover HelloHR employees have the possibility to change a label, providing feedback to the AWS system to improve future classification.

To support this flow, a question classifier needs to be developed, represented as 'best model' in figure 11. To do this, a target dataset is

Figure 10: *Flowchart of the Proof of Concept in practice.*



created as discussed in section 5.1 and several experiments are performed, which will be discussed in section 6. For the experiments, the SageMaker supported components of the TFX framework return. In the Proof of Concept, these components will be configured and tested on a local device instead of on SageMaker, due to limited capacity available to configure the AWS environment and SageMaker module. The TFX components fully tested for the methods discussed in 3.2.1 'Transformation' and 3.2.2 'Classification' through the PoC are implementations of the *Data Transformation*, *Trainer* and *Tuner* components. These are tested using an implementation of the *Model Evaluation and Validation* component. Whilst some idea exists for the implementation of data validation in the context of new labels to be added by HelloHR and potential changes to the vocabulary of the dataset, impacting the solution, these were not extensively tested, but will be discussed in section 8 'Discussion'.

The PoC design abstracts process already extensively explained in the full solution design as 'other processes' and assumes the same for TOPdesk. The core difference with the full solution is the scope of the solution and the ability to automatically retrain the best model with recent data.

In scope are twelve labels, of which one a negative label representing that no question has been classified. This means that the solution can potentially answer eleven types of questions, using the model that will result from various experiments. The creation of the target dataset from an initial dataset has been extensively described in previous section 5.1 'Target Dataset', and would represent the Data Ingestion component of the TFX framework.

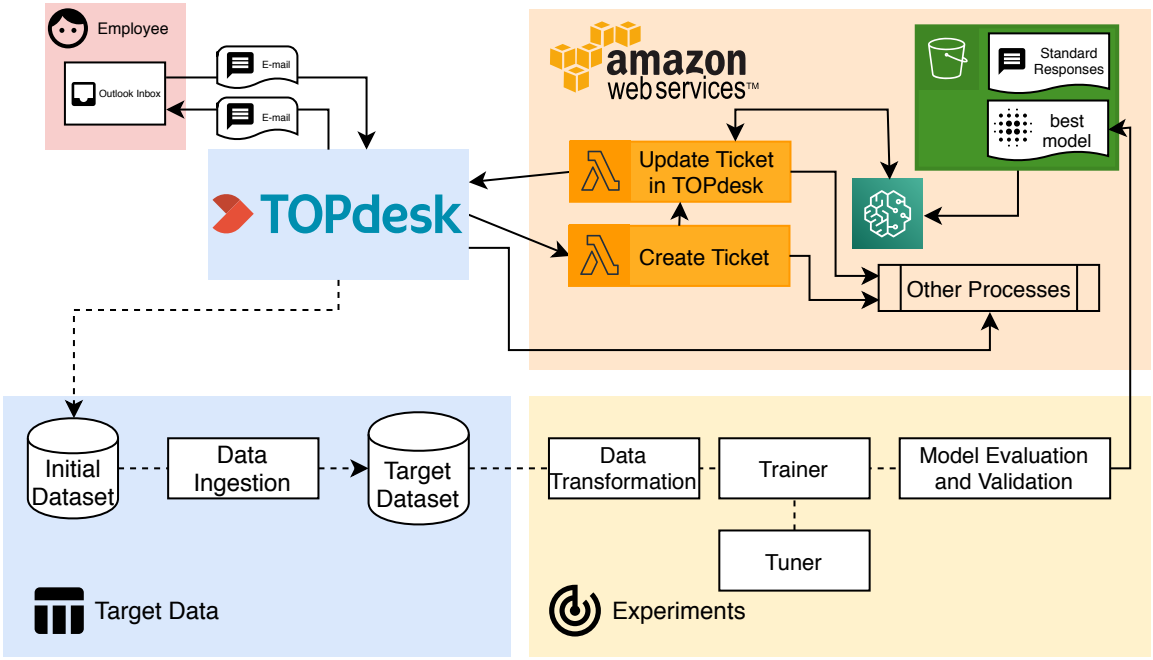
6 Experimental Setup

In this section, experiments are described related to the data transformation, training & tuning and model evaluation and validation components of the TFX model, to achieve the best model for the Proof of Concept. Moreover, metrics are set-up that allows for the derivation of the performance of the Proof of Concept, the potential quantifiable effects on operations of the HR department & expected timeframe of the investment.

6.1 Experiment 1: Data, Transforms, Model

In section 3 'Background', several methods for transforming data have been described, as well as several supervised machine learning models that can create classification boundaries based on transformed data and labels. In section 5.1

Figure 11: *The design of the Proof of Concept. A serving model is developed that can function in the context of the full solution as shown in figure 7.*



'Target Dataset', these labels were acquired from clean data, and moreover, different types of n-grams were used to create separate columns of clean data. Due to limited computation power, no experiments are done with oversampling and undersampling, as this would even further increase the number of experiments. Applying the no free lunch theorem, no perfect optimization solution exists that fits all types of datasets [84]. Therefore, all variations of data, transformation and model are evaluated. An overview of the different variables in the experiment to be tested can be found in table 4. In total, 93 tests were done to find the optimal configuration of N-grams, Transformation and Model for each classification model. For the Bi-LSTM, weights of an embedding layer were either initialised using learned weights from FastText or Word2Vec or not. Due to the subpar performance of FastText and Word2Vec and due to the large training time required to train the Bi-LSTM, no TF-IDF weighted word embeddings were used. An overview of all hyperparameters for this experiment can be found in Appendix D 'Hyperparameters'. These experiments are done to select the best data input and transformation for each model.

Table 4: *Overview of methods for the experiments.*

N-gram	Transformation	Model
Unigram	Count	OvO SVM
Bigram	TF-IDF	OvR SVM
Trigram	Word2Vec	RF
	Word2Vec+TF-IDF	XGBoost
	FastText	
	FastText+TF-IDF	
	Doc2Vec	

N-gram	Embedding	Model
Unigram	No Weights	Bi-LSTM
Bigram	FastText	
Trigram	Word2Vec	

As the SVM was already tested thoroughly as it was used during labelling, hyperparameters

were already tuned during this experiment. Tuning of the RandomForest (RF) model was done by first performing a random grid-search on a wide range of hyperparameters and afterwards a full grid-search based on the random grid-search. No tuning was done for the XGBoost and Bi-LSTM models due to computational limitations.

Moreover, the Bi-LSTM is an extremely simple implementation of the method, having only a single embedding layer and Bi-LSTM layer with a softmax output. This limited implementation was chosen due to the lack of computing power, which will be resolved when moving the future solution to AWS.

6.2 Experiment 2: Data Volume

To test the necessary data required to train the model and to see if more data can be added to further improve the performance of the model, an experiment was done with the tuned variants of the SVM and RandomForest. Due to lacking performance in experiment 1, and overall high required computation time, the Bi-LSTM and XGBoost are left out of scope for this experiment.

During this experiment, the models are trained on an increasing amount of data, expanding over time. Here, the most recent 6 months are kept as a test set. With the rest of the data, starting from the 7th most recent month, an estimator is trained and evaluated, after which the next month is added to the training data, training a new estimator and evaluating again. This is done until the 17th month, leading to 11 experiments. The 17th month has been chosen as a cut-off point, as the data volumes in the months prior to 2018 are lower in volume compared to after 2018, as can be seen in figure 2. During these experiments, the label *ns_greenwheels* was removed, as this label only existed in the test set. The value counts of the labels for the test set are displayed in 5.

Label	**Count**
NaN	16251
sf_loginissue	799
yf_loginissue	268
wni_request	108
bp_loginissue	104
expiring_leave	90
yf_process_issue	78
change_personal_details	78
adjust_leave	53
changepoint	53
lms_loginissue	34

Table 5: *Test set for Experiment 2.*

The goal of this experiment is to discover if the addition of more data will improve the model performance. If this is not the case, other methods need to be found in order to improve the overall performance of the model. Moreover, since the General Data Protection Law dictates that data can only be kept for a certain amount of time, the impact of a limiting data retention policy can be assessed [85].

6.3 Evaluation Criteria

In section 3.2.4 'Evaluation metrics', several evaluation metrics have been introduced. In this thesis, a variant of the F1 score is used, $F_{0.5}$. The $F_{0.5}$ score can be calculated using the general formula for the calculation of the harmonic mean between precision and recall, described as F_β, shown in equation 7.

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall} \quad (7)$$

Here, a β of 0.5 used in $F_{0.5}$ would favour precision over recall in the resulting score. Choosing a β of 2, for example, would result in a bias towards recall in the final scoring. $F_{0.5}$ is chosen over F_1 and F_2, as a wrong answer sent to the employee is a worse outcome than not providing an answer.

In case a wrong answer is given, for which the likelihood is higher with high precision, the em-

ployee will lose time by having to understand that a mistake has been made and reply to the wrong answer, having to wait again for a response, this time by an employee. Moreover, this is detrimental for the employee experience, as low quality of service is provided through the new solution.

In case no answer is given, by classifying the problem as 'NaN', nothing changes for the employee or the HelloHR employee, except for the fact the HelloHR employee can adjust the label such that the question might be recognized in the future. Here, the status quo is met.

In this context, the 'NaN' label forms a negative class for the whole classification problem. As a consequence, the precision and recall of the 'NaN' label are not taken into account when calculating the $F_{0.5}$ score.

In addition to choosing $F_{0.5}$ as the evaluation criteria, a choice has to be made between the calculation of $F_{0.5}$ on a macro or micro level. In both cases, the score for the 'NaN' class is ignored.

In the evaluation of the model performance in the experiments, the micro $F_{0.5}$ score is chosen as a metric. This micro score allows for scoring on the true impact of the model, answering as many employee questions correctly. The more balanced approach macro will ensure that the model performs optimally across all classes, but will not necessarily result in the most optimal impact.

To validate the results of the model, shuffled[4] stratified k-fold cross-validation was used. For each fold, a $F_{0.5}$ score was calculated using the holdout test set. Using these scores, the mean and the standard error of the $F_{0.5}$ score was calculated, allowing for the calculation of a confidence interval. For each experiment, the confidence of 95% was calculated assuming a normal distribution of the $F_{0.5}$ score.

7 Results

In this section, the results of the experiments as described in section 6 'Experimental Setup' are described. First, the outcomes of the experiments are presented. Second, the operational impact is calculated in terms of return on investment and effects on operations.

7.1 Experiment 1

In experiment 1, several configurations of data, transform and model were tested using the evaluation metric $F_{0.5}$ validated using shuffled stratified 5-fold cross validation resulting in a confidence interval around the mean $F_{0.5}$ score for 95% ($\alpha = 0.05$). The results are presented in table 6. A more detailed result overview including metrics per class can be found in Appendix E.

Here, a selection area is given using the confidence interval of the highest mean $F_{0.5}$ for each model. Model configurations for which the confidence intervals overlap with the selection area should be considered for the best model. As a result, most models have several configurations within this selection area. The most practical configuration is chosen. For the n-grams, the unigrams represent the most practical option. For the vectorizers, the FastText embeddings represent the most practical option, as word embeddings for out of vocabulary words can be estimated using character level n-grams.

The difference between unigrams, bigrams and trigrams was insignificant across experiments. Unigram input data is seen as the best across all models for this dataset. Doc2Vec performed very poorly across all models, most likely due to the large imbalance across class labels in the dataset, resulting in poor features for the Doc2Vec word embedding model which relies on these labels for contextual information.

Support Vector Machine

For SVM, the decision between the best transformer is split between the TF or the TF-IDF vectorizer. The difference in practicality is non-existent between TF and TF-IDF vectorization. However, TF-IDF allows for further tuning of hyperparameters compared to TF, such as the exclusion of certain terms based on their document frequency. Therefore, TF-IDF is chosen as the most optimal transformer for SVM.

[4] To maintain consistency across experiments, the seed generated by numpy.random.seed(500) was used

Table 6: *Overview of Experiment 1 Results. Selection area is arced, best score is boldfaced.*

One vs Rest SVM	**Unigram**	**Bigram**	**Trigram**
Doc2Vec	0.417 +/- 0.009	0.431 +/- 0.023	0.433 +/- 0.027
Word2Vec	0.804 +/- 0.003	0.801 +/- 0.005	0.800 +/- 0.017
Word2Vec + TF-IDF	0.760 +/- 0.020	0.762 +/- 0.014	0.759 +/- 0.011
FastText	0.787 +/- 0.013	0.790 +/- 0.017	0.790 +/- 0.023
FastText + TF-IDF	0.753 +/- 0.013	0.751 +/- 0.013	0.760 +/- 0.013
TF-IDF	0.874 +/- 0.016	**0.878 +/- 0.016**	0.876 +/- 0.011
Count	0.861 +/- 0.018	0.860 +/- 0.015	0.864 +/- 0.012

One vs One SVM	**Unigram**	**Bigram**	**Trigram**
Doc2Vec	0.443 +/- 0.020	0.454 +/- 0.014	0.464 +/- 0.028
Word2Vec	0.808 +/- 0.010	0.805 +/- 0.012	0.800 +/- 0.013
Word2Vec + TF-IDF	0.771 +/- 0.007	0.766 +/- 0.024	0.764 +/- 0.008
FastText	0.799 +/- 0.019	0.795 +/- 0.012	0.799 +/- 0.012
FastText + TF-IDF	0.762 +/- 0.013	0.765 +/- 0.020	0.762 +/- 0.016
TF-IDF	**0.881 +/- 0.006**	0.879 +/- 0.015	0.878 +/- 0.007
Count	0.857 +/- 0.014	0.859 +/- 0.019	0.862 +/- 0.016

RandomForest	**Unigram**	**Bigram**	**Trigram**
Doc2Vec	0.139 +/- 0.016	0.146 +/- 0.021	0.146 +/- 0.027
Word2Vec	0.760 +/- 0.015	0.765 +/- 0.022	**0.766 +/- 0.019**
Word2Vec + TF-IDF	0.758 +/- 0.020	0.765 +/- 0.005	0.766 +/- 0.023
FastText	0.744 +/- 0.027	0.745 +/- 0.015	0.742 +/- 0.009
FastText + TF-IDF	0.748 +/- 0.013	0.753 +/- 0.018	0.752 +/- 0.017
TF-IDF	0.562 +/- 0.012	0.548 +/- 0.006	0.549 +/- 0.011
Count	0.426 +/- 0.009	0.421 +/- 0.010	0.426 +/- 0.010

XGBoost	**Unigram**	**Bigram**	**Trigram**
Doc2Vec	0.245 +/- 0.027	0.266 +/- 0.045	0.258 +/- 0.028
Word2Vec	0.677 +/- 0.018	0.670 +/- 0.034	**0.681 +/- 0.011**
Word2Vec + TF-IDF	0.673 +/- 0.016	0.666 +/- 0.009	0.678 +/- 0.022
FastText	0.632 +/- 0.033	0.647 +/- 0.017	0.656 +/- 0.012
FastText + TF-IDF	0.654 +/- 0.012	0.658 +/- 0.011	0.657 +/- 0.018
TF-IDF	0.595 +/- 0.028	0.578 +/- 0.037	0.577 +/- 0.012
Count	0.566 +/- 0.021	0.540 +/- 0.038	0.539 +/- 0.025

Bi-LSTM	**Unigram**	**Bigram**	**Trigram**
Word2Vec	0.480 +/- 0.054	0.508 +/- 0.046	0.469 +/- 0.076
FastText	0.469 +/- 0.100	0.442 +/- 0.066	0.474 +/- 0.036
No Weights	**0.687 +/- 0.074**	0.651 +/- 0.054	0.671 +/- 0.109

Aside from a transformation technique, a choice has to be made between the multiclass training strategies One-Versus-One and One-Versus-Rest. Here, the performance of both is not significantly different. The training complexity of One-Versus-Rest, however, is equal to the number of classes, whereas the training complexity of a One-Versus-One strategy equals $N(N-1)/2$, where N is the number of classes. Therefore, One-Versus-Rest is the more practical solution due to less required computational power. The One-Versus-Rest strategy is chosen as the default SVM training strategy for this Proof of Concept.

Random Forest Classifier
The random forest classifier seems to perform best on word embedding based transformation techniques. As stated before, FastText is seen as the most practical of all word embedding options, allowing for out of vocabulary words to be embedded and used as input for the classifier. Therefore, it was chosen to use FastText as an optimal transformer for the RandomForest for this Proof of Concept.

XGBoost
Similar to Random Forest, word embedding based transformation techniques are significantly better than others as input for the XGBoost algorithm. Unsurprising, since both models are trained based on the concept of decision trees. Interesting, however, is that Word2Vec embedding seems to score significantly better than FastText embeddings. With the difference between TF-IDF weighted Word2Vec embeddings and regular Word2Vec embeddings being non-significant, the less complex transformation technique Word2Vec is chosen as the best transformer for XGBoost for this Proof of Concept.

Bi-LSTM
Overall, the Bi-LSTM implementation performed best with no-weights, most likely due to the increased amount of trainable parameters that come with a trainable embedding layer. Due to computational limits, not enough iterations could be made to evaluate the learning curve of the model to choose a desirable amount of epochs, which could potentially improve the performance of the unweighted Bi-LSTM.

Whilst the performance of the Bi-LSTM using FastText and Word2Vec embeddings seems lacklustre, the unweighted results show promise for the use of sequence-based language models to improve beyond the Proof of Concept.

From the results presented in Appendix E, the SVM classifier performs the best with regards to the micro $F_{0.5}$ score.

7.2 Experiment 2

From the results presented in figure 12, we can deduce that more data won't necessarily make the model better. Whereas the SVM might still slightly improve by adding more data, the RandomForest will not improve by adding more data points. These results are not surprising, as, in contrast to deep learning models, SVM and RandomForest are not algorithms who are known to benefit from massive amounts of data.

Figure 12: *Micro $F_{0.5}$ score for One vs Rest SVM and a RandomForest for a cumulative amount of months of training data.*

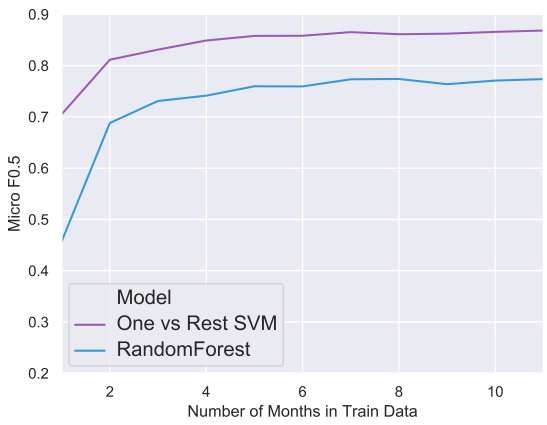


Figure 13: *Macro $F_{0.5}$ score for One vs Rest SVM and a RandomForest for a cumulative amount of months of training data.*

As these results are based on a test set that takes only a sample of the potential question-types that could be answered, it tends to be biased towards the majority class within the labels that are in scope, in this case, *sf_loginissues*. It is expected This, however, makes these type of models extra suitable for application within the context of HR, or other organisation specific QA based datasets. In general, these are smaller than consumer-facing QA datasets, where deep learning might outshine these models. Finally, as SVM benefits mostly from adding more difficult data points, as these can serve as support vectors, correcting of misclassifications by HR employees are still expected to improve the current state of the Micro F0.5 score.

7.3 Classification Examples

In this section, some example behaviour is shown for the selected best classifier + transform of Unigram, TF-IDF and One-vs-Rest SVM. First, as the classifier is trained on binary labels and questions where multiple question classes occur are labelled as 'NaN'. The expected behaviour is thus that if an employee request contains multiple questions, the classification of 'NaN' will be given. Examples for this behaviour are given in table 7.

The expected behaviour is performed except for the last example. Benefitplaza in the subject is filtered by the Part of Speech tagger, leading to a classification 'yf_loginissue', most likely because youforce is now mentioned twice whereas benefitplaza is only mentioned once after cleaning. This shows the sensitivity of the solution to incorrect grammar and language use .

Table 7: *Behaviour combined question.*

Subject	Youforce en benefitplaza
Request	Hoi, ik heb tot heden nog geen inloggegevens voor Youforce en Benefitplaza ontvangen. Kunnen jullie deze toesturen?
Predicted Label	NaN
Subject	Youforce
Request	Hoi, ik heb tot heden nog geen inloggegevens voor Youforce ontvangen. Kunnen jullie deze toesturen?
Predicted Label	yf_loginissue
Subject	Benefitplaza
Request	Hoi, ik heb tot heden nog geen inloggegevens voor Benefitplaza ontvangen. Kunnen jullie deze toesturen?
Predicted Label	bp_loginissue
Subject	Youforce Benefitplaza
Request	Hoi, ik heb tot heden nog geen inloggegevens voor Youforce en Benefitplaza ontvangen. Kunnen jullie deze toesturen?
Predicted Label	yf_loginissue

Another interesting behaviour to investigate is the reaction to a change in the vocabulary of end-users. Recently, VodafoneZiggo has changed the system in which employees can declare expenses. Formerly, this was called 'Benefitplaza', which is currently captured under the label *bp_loginissue*. Now, this system is called 'benefit connect'.

Table 8: *Behaviour changed vocabulary.*

Subject	Inloggen benefitplaza
Request	Hoi, ik ben mijn wachtwoord kwijt voor benefitplaza. Ik wil mijn declaraties doen.
Predicted Label	bp_loginissue

Subject	Wachtwoord kwijt
Request	Hoi, ik ben mijn wachtwoord voor het systeem waarin ik declaraties wil doen.
Predicted Label	NaN

Subject	Wachtwoord kwijt
Request	Hoi, ik ben mijn wachtwoord voor benefit connect kwijt. Ik wil mijn declaraties doen.
Predicted Label	NaN

Here we can see that the system does not react well to changed system names out of the box. This behaviour should be further explored in practice. Since these labels will be corrected and can change over time, it has to be seen how many tickets with the new system name or vocabulary are required to improve the serving model for a specific label affected by the vocabulary change.

The final behaviour to be investigated is what will occur if unrelated noise is added to a question. As the SVM tends to classify requests with multiple questions as 'NaN'. An example of the behaviour of the model in this case is shown in table 9

Table 9: *Behaviour unrelated noise.*

Subject	Bovenwettelijk verlof
Request	Beste, ik vraag me af waarom er in youforce staat dat mijn bovenwettelijke verlof per juli te vervallen komt. Kan dit hersteld worden?
Predicted Label	expiring_leave

Subject	Bovenwettelijk verlof
Request	Beste, Ik vraag me af waarom er in youforce staat dat mijn bovenwettelijke verlof per juli te vervallen komt. Ik wil graag drie weken op vakantie met mijn familie, wat nu niet gepland kan worden omdat ik mijn verlof opeens kwijt ben. Kan dit hersteld worden?
Predicted Label	expiring_leave

Subject	Bovenwettelijk verlof
Request	Beste, Ik vraag me af waarom er in youforce staat dat mijn bovenwettelijke verlof per juli te vervallen komt. Kan dit hersteld worden? Sent by my iphone Please do not print this e-mail to save the environment. Print deze mail niet om het milieu te besparen.
Predicted Label	NaN

The model seems to perform fine with noise that is slightly related to the question at hand, but predict a NaN value if there is too much noise in the ticket. Whilst extensive cleaning steps have been taken to prevent this behaviour from occuring, improvements in the cleaning procedure that better extract the question from the text will be expected to improve the performance of the classifier.

7.4 Effect on operations

Based on the performance of the SVM, approximately 81% of the questions that are in the scope of the Proof of Concept can be answered with an error rate of approximately 13%. Based on the assumption that an HelloHR employee would spend around 5 minutes on answering a ticket, the fact that the questions in scope cover 8.9% of the historical ticket load, an estimated 7.2% of the total e-mail related ticket volume can be automated. Based on the mean amount of e-mail based tickets in the past months shown in 2, 273 tickets would be automatically answered by the Proof of Concept. This would save the operations of the HR department an estimated $22\frac{3}{4}$ hours every month, or about 0.569 FTE.

The cost of development of the full solution is estimated at around € 20,000.-, with operational costs per month estimated at € 44 excl. btw, including the cost of development instances based on the full solution design. Thus far, costs were made for € 1800.- by hiring two consultants to create the TOPdesk API. Estimating 1 FTE per month at a conservative € 2600.-, the solution would have a return on an investment after:

$$\frac{21{,}800eur}{(0.569\times 2600eur/mnth)-44eur/mnth} \approx 15{,}19 \text{ months}$$

8 Discussion

The goal of this thesis was to design a question answering system that could respond to operational questions at the HR department to reduce workload and response time, to improve the operations of the HR department. Throughout this study, a full solution design has been developed that would allow a question answering system to become operational at the client. Moreover, a Proof of Concept was developed for the client, resulting in a very basic solution that would break even after about 15 months. This solution achieves the low error rate desired by the client, whilst maintaining the overall impact.

8.1 Benefits & Costs

Whilst the desired error rate has been achieved, the potential impact of the question answering system can be expanded by improving the number of labels in the dataset. This expansion is already described in the full solution design and will allow for the identification of more labels and growth of the scope of the model. Moreover, advances could be made by tuning the XGBoost classifier and expanding the Bi-LSTM Neural Net, as these options could not be explored during this thesis due to limited time and computational power available at the client.

Whilst the final machine learning pipeline is fully automated, allowing for higher quality models based on new, recent data, functional maintenance is required to keep the standard responses up to date and to add new labels to the scope of the system. It is expected that an update of the whole system around every 6 months will allow for an expansion of the scope of the QA system, improving overall performance and impact, as well as the operational value created.

Aside from operational benefits, several strategic benefits have also been identified. Being able to track the type of questions employees are asking allows for targeted intervention through communication, changes in HR policies or changes in HR practices. This provides insights into the HR department that wasn't available before.

Overall, both the operational value but also the strategic value of artificial intelligence as well as Cloud technologies for the HR department has emerged during this study. Whilst the solution has been developed in a client context, the generalizability of the methods used to develop the final solution and the proof of concepts to other contexts or industries is quite high. Of course,

the same results using similar parameters for a different dataset cannot be guaranteed, as dictated by the No Free Lunch Theorem [84]. It is expected however that applying similar techniques as were applied in this study to a question-answering context, a question answering system can be built.

8.2 Sensitivity

The sensitivity of the question answering system to external changes has been briefly assessed. When asking multiple questions or mentioning too many unrelated words unrelated to the problem at hand after asking the question, the system will almost certainly classify the question as 'NaN', potentially missing an employee question.

Moreover, whenever the context of label changes, it is expected that retraining using new data entries with these context changes embedded are required. After retraining on new data with the changed context, the model should slowly converge to regular performance again. An example of changed context would be the replacement of an IT solution or component in the scope of the login issues or the change in nomenclature for certain services, such NS Greenwheels, which is closely related to a service name.

8.3 Legislation

Whilst the solution design works as intended, the data used in this case study is sensitive according to the General Data Protection Law [85]. This can have implications on the amount of time training data can be stored on AWS for automatic training if the training data can be stored at all. Experiment 2 showed that with a limited amount of data, acceptable results could be achieved. With more data however, the solution would perform better on a wider variety of questions. Moreover, to enable more advanced models such as a possible expansion of the Bi-LSTM Neural Net, more data is expected to yield better performance.

The solution design and proof of concept are currently at the review at the privacy office of the client to assess if an implementation is legally allowed. However, since the solution service a beneficial purpose for both HR employees as well as employees with HR-related questions, there is grounded reason for the legality of the solution according to the GDPR legislation [85], as this legislation was not meant to disrupt beneficial innovations.

9 Conclusion & Future Outlook

As the potential value for the HR department as a result of Industry 4.0 technologies, such as Cloud Technologies and Artificial Intelligence, has become evident from this study and several others in terms of operational and strategic value, the demand for similar Industry 4.0-type innovations within HR is expected to rise. This study can serve as a guideline for what is possible within the context of an organisation and especially the HR department.

The HR department does not have the amount of data a customer serving business unit might have, but some data might be big enough to extract value from. This study has shown that from an unlabeled dataset, an operation ready QA system can be developed. Moreover, the full solution has been described in terms of high-level machine learning components and existing cloud services using AWS. All in all, this study provides a case study that provides insights to HR practitioners who are not aware of the possibilities of Cloud Solutions and the enabling features for Artificial Intelligence in organisations, supported with operational benefits in the context of the HR department.

Aside from an improved Bi-LSTM Neural Net or tuned XGBoost model, techniques that remained unused through this study could serve for new experiments that can improve the performance of the final solution, such as various techniques of resolving imbalance or the BERT and ELMo word embedding models. Moreover, the data to develop the model could be expanded to include documentation or Q&A type of com-

munications of an organisation to improve the quality of word embedding techniques.

Another thing element that might have impacted the research is the labelling technique. For this study, an SVM, as these are expected to perform best on imbalanced data and have a straightforward hyperparameter that allows for a wider range of errors to be made. Whilst the labels have been expanded with the use of RandomForest after an initial run of Experiment 1, the False Positives of XGBoost and Bi-LSTM are yet to be evaluated. Labelling more data could affect the results of the tests by correcting incorrect False Positives, expected to improve precision.

10 Acknowledgement

I would like to thank my supervisors at both the University of Twente and VodafoneZiggo for supporting me and inspiring me to move beyond just developing a proof of concept, but a solution that could potentially have a huge impact on the future of HR operations when implemented correctly. My thanks also go out to all colleagues who in some way supported this project, either by providing expert advice on certain systems or by having discussions with me about the components of the solution design and potential value for the HR department. My motivation was largely derived from positive and especially inspired experiences had by people whom I explained the project too. Thanks!

References

[1] A. Ustundag and E. Cevikcan, *Industry 4.0: managing the digital transformation*. Springer, 2017.

[2] H. Kagermann, "Change through digitization—value creation in the age of industry 4.0," in *Management of permanent change*, pp. 23–45, Springer, 2015.

[3] T. Stock and G. Seliger, "Opportunities of sustainable manufacturing in industry 4.0," *Procedia Cirp*, vol. 40, pp. 536–541, 2016.

[4] M. Onik, M. H. Miraz, and C.-S. Kim, "A recruitment and human resource management technique using blockchain technology for industry 4.0," 2018.

[5] S. Shamim, S. Cang, H. Yu, and Y. Li, "Management approaches for industry 4.0: A human resource management perspective," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 5309–5316, IEEE, 2016.

[6] D. Angrave, A. Charlwood, I. Kirkpatrick, M. Lawrence, and M. Stuart, "Hr and analytics: why hr is set to fail the big data challenge," *Human Resource Management Journal*, vol. 26, no. 1, pp. 1–11, 2016.

[7] J. H. Marler and J. W. Boudreau, "An evidence-based review of hr analytics," *The International Journal of Human Resource Management*, vol. 28, no. 1, pp. 3–26, 2017.

[8] S. Van den Heuvel and T. Bondarouk, "The rise (and fall?) of hr analytics: A study into the future application, value, structure, and system support," *Journal of Organizational Effectiveness: People and Performance*, vol. 4, no. 2, pp. 157–178, 2017.

[9] VodafoneZiggo, "Jaarverslag 2018, onze organisatie." `https://www.jaarverslagvodafoneziggo.nl/verslag/onze-organisatie`, 2018. Accessed: 2019-05-06.

[10] VodafoneZiggo, "Jaarverslag 2018, strategy and financial results." `https://www.annualreportvodafoneziggo.com/report/strategy-and-financial-results/strategy-and-financial-results`, 2018. Accessed: 2019-07-11.

[11] VodafoneZiggo, "Jaarverslag 2018, strategie en financiële resultaten." `https://www.jaarverslagvodafoneziggo.nl/verslag/strategie-en-financiele-resultaten/financiele-resultaten`, 2018. Accessed: 2019-05-06.

[12] Advanced Television, "Vodafone, liberty global welcome merger clearance."

[13] P. B. Brandtzaeg and A. Følstad, "Why people use chatbots," in *International Conference on Internet Science*, pp. 377–392, Springer, 2017.

[14] E. Devaney, "Sate of conversational marketing 2017." `https://www.drift.com/blog/state-of-conversational-marketing/`, 2017. Accessed: 2019-03-04.

[15] A. Deshpande, A. Shahane, D. Gadre, M. Deshpande, and P. M. Joshi, "A survey of various chatbot implementation techniques," *International Journal of Computer Engineering and Applications*, vol. 11, 2017.

[16] M. Kuperstein, "Speech recognition dialog management," Jan. 15 2009. US Patent App. 11/629,034.

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[18] J. W. White, "Method and apparatus for generating transactions and a dialog flow manager," Sept. 5 2000. US Patent 6,115,711.

[19] L. Ciechanowski, A. Przegalinska, M. Magnuski, and

P. Gloor, "In the shades of the uncanny valley: An experimental study of human–chatbot interaction," *Future Generation Computer Systems*, vol. 92, pp. 539–548, 2019.

[20] E. Brill, S. Dumais, and M. Banko, "An analysis of the askmsr question-answering system," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 257–264, Association for Computational Linguistics, 2002.

[21] R. F. Simmons, "Natural language question-answering systems: 1969," *Communications of the ACM*, vol. 13, no. 1, pp. 15–30, 1970.

[22] L. Hirschman and R. Gaizauskas, "Natural language question answering: the view from here," *natural language engineering*, vol. 7, no. 4, pp. 275–300, 2001.

[23] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[24] R. Wieringa and A. Moralı, "Technical action research as a validation method in information systems design science," in *International Conference on Design Science Research in Information Systems*, pp. 220–238, Springer, 2012.

[25] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: an automatic question-answerer," in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pp. 219–224, ACM, 1961.

[26] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III, "A neural network for factoid question answering over paragraphs," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 633–644, 2014.

[27] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

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

[29] A. Patra and D. Singh, "A survey report on text classification with different term weighing methods and comparison between classification algorithms," *International Journal of Computer Applications*, vol. 75, no. 7, 2013.

[30] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[31] A. Graves, "Supervised sequence labelling with recurrent neural networks," 2012.

[32] Microsoft, "Skype for business." `https://www.skype.com/en/business/`, 2019. Accessed: 2019-07-12.

[33] Microsoft, "Enterprise social network yammer." `https://products.office.com/en/yammer/yammer-overview`, 2019. Accessed: 2019-07-12.

[34] TOPdesk, "Topdesk documentation." `https://developers.topdesk.com/documentation/index.html`, 2019. Accessed: 2019-07-12.

[35] TOPdesk, "Topdesk settings to the web." `https://developers.topdesk.com/documentation/index.html`, 2019. Accessed: 2019-23-09.

[36] Y. Chen, B. Dong, Y. Shen, W. Zhenglong, and X. Liu, "Question answering framework," Dec. 15 2015. US Patent 9,213,771.

[37] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[38] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[40] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *arXiv preprint arXiv:1607.05368*, 2016.

[41] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[42] X. Wang, J. Wu, D. Zhang, Y. Su, and W. Y. Wang, "Learning to compose topic-aware mixture of experts for zero-shot video captioning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8965–8972, 2019.

[43] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014.

[44] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018.

[45] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[46] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, ACM, 1992.

[47] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," *Kernel methods in computational biology*, vol. 47, pp. 35–70, 2004.

[48] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[49] sci-kit learn, "Support vector machines." `https://scikit-learn.org/stable/modules/svm.html#kernel-functions`, 2019. Accessed: 2019-09-21.

[50] Y. Goldberg and M. Elhadad, "splitsvm: fast, space-

efficient, non-heuristic, polynomial kernel computation for nlp applications," in *Proceedings of ACL-08: HLT, Short Papers*, pp. 237–240, 2008.
[51] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
[52] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[53] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.
[54] M. Robnik-Šikonja, "Improving random forests," in *European conference on machine learning*, pp. 359–370, Springer, 2004.
[55] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
[56] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, ACM, 2016.
[57] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*, 2016.
[58] B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, vol. 31, no. 1-4, pp. 107–123, 2000.
[59] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
[60] J. Alqatawna, H. Faris, K. Jaradat, M. Al-Zewairi, and O. Adwan, "Improving knowledge based spam detection methods: The effect of malicious related features in imbalance data distribution," *International Journal of Communications, Network and System Sciences*, vol. 8, no. 05, p. 118, 2015.
[61] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 04, pp. 687–719, 2009.
[62] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.
[63] R. C. Prati, G. E. Batista, and M. C. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior," in *Mexican international conference on artificial intelligence*, pp. 312–321, Springer, 2004.
[64] N. Japkowicz, "Concept-learning in the presence of between-class and within-class imbalances," in *Conference of the Canadian society for computational studies of intelligence*, pp. 67–77, Springer, 2001.
[65] C. Drummond, R. C. Holte, *et al.*, "C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling," in *Workshop on learning from imbalanced datasets II*, vol. 11, pp. 1–8, Citeseer, 2003.
[66] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.
[67] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
[68] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, IEEE, 2008.
[69] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.
[70] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, *et al.*, "Tfx: A tensorflow-based production-scale machine learning platform," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1387–1395, ACM, 2017.
[71] AWS, "Aws lamda." `https://aws.amazon.com/lambda/`, 2019. Amazon. Accessed 2019-09-23.
[72] AWS, "Amazon api gateway." `https://aws.amazon.com/api-gateway/`, 2019. Amazon. Accessed 2019-09-23.
[73] AWS, "Aws simple queue service." `https://aws.amazon.com/sqs/`, 2019. Amazon. Accessed 2019-09-23.
[74] AWS, "Amazon s3." `https://aws.amazon.com/s3/`, 2019. Amazon. Accessed 2019-09-23.
[75] AWS, "Amazon dynamodb." `https://aws.amazon.com/dynamodb/`, 2019. Amazon. Accessed 2019-09-23.
[76] AWS, "Aws glue." `https://aws.amazon.com/glue/`, 2019. Amazon. Accessed 2019-09-23.
[77] AWS, "Aws sagemaker." `https://aws.amazon.com/sagemaker/`, 2019. Amazon. Accessed 2019-09-23.
[78] AWS, "Amazon step functions." `https://aws.amazon.com/step-functions/`, 2019. Amazon. Accessed 2019-09-23.
[79] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.
[80] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
[81] Explosion AI, "Spacy - dutch." `https://spacy.io/models/nl`, 2019. Accessed: 2019-09-24.
[82] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.
[83] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
[84] D. H. Wolpert, W. G. Macready, *et al.*, "No free lunch theorems for optimization," *IEEE transactions*

on evolutionary computation, vol. 1, no. 1, pp. 67–82, 1997.
[85] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.
[86] Python Software Foundation, "Python release 3.7.2." `https://www.python.org/downloads/release/python-372/`, 2019. Accessed: 2019-09-24.
[87] G. Diaz, "Stopwords nl." `https://github.com/stopwords-iso/stopwords-nl`, 2019. Accessed: 2019-09-24.
[88] Scikit-learn developers, "Svc."
[89] Scikit-learn developers, "Randomforestclassifier."
[90] Scikit-learn developers, "Multiclass and multilabel algorithms."
[91] XGBoost developers, "Xgboost python api reference."
[92] Keras developers, "Keras documentation."
[93] Scikit-learn developers, "Countvectorizer."
[94] Scikit-learn developers, "Tfidfvectorizer."

Appendices

A TOPdesk-AWS Integration

In this section, a demo is shown showcasing the interaction between TOPdesk and AWS in context of the proof of concept shown in figure 11. This demo shows the perspective of an HR employee, who will only see a ticket if:

1. The QA system classified a question as 'NaN'
2. An employee sends another request after a question has been answered by the QA system

Moreover, the API calls made between the systems are described. This shows the decided protocols for communication between the two systems, including what type of data is sent back and forth under which fields.

A.1 Demo

This demo will be shown through screenshots of the system, highlighting specific elements of interest, supported by schematics that illustrate the communication protocol designed for the two systems to communicate.

In the standard overview, presented in figure 14, several actions have been performed. Here, the AI has provided an answer to the question given by the employee, which was unsatisfying for the employee, as the employee has replied to the request and the ticket has arrived at HelloHR.

In figure 14, several elements of the interface have been marked. These have the following meaning:

1. The question asked by the employee.
2. This is the answer given by the QA system. As the QA system is not installed, a default response is returned by AWS 'Antwoord' (answer).
3. Together with an answer, AWS also changes the Category and Subcategory. In this case, these were set to 'HR applicaties' and 'Youforce' respectively.
4. A hidden message is added to the ticket asking the employee to check the tab 'extra information' in order to change a label, as this label is most likely wrong, given a response has been received by the employee.
5. The employee can add an action to the ticket in this field, as is done in the current process.
6. When the label is changed, a request is sent to AWS to change the label in the DynamoDB table for that specific tableID, as shown in 7.
7. The tab 'Extra information' can be opened to change the label of the ticket, assigned by AWS.

To change a label, one opens the 'Extra information' tab, giving the overview as presented in figure 15. Here, one can select a label through a drop-down menu.

1. The old label
2. The dropdown menu from which a new label can be selected. The correct label is highlighted.

Figure 14: *The demo overview from an employee perspective.*

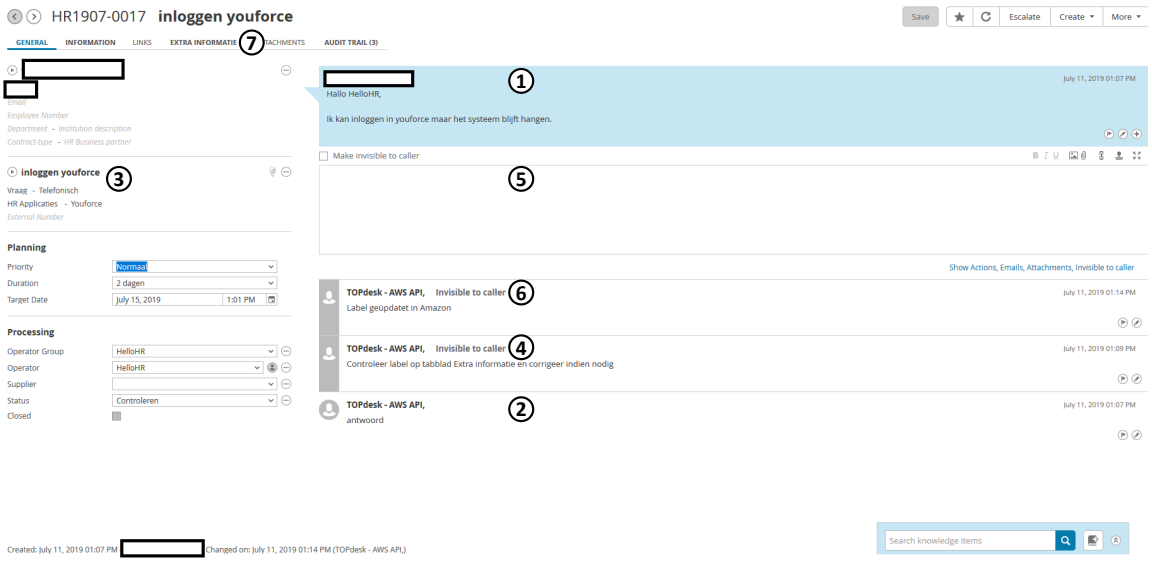
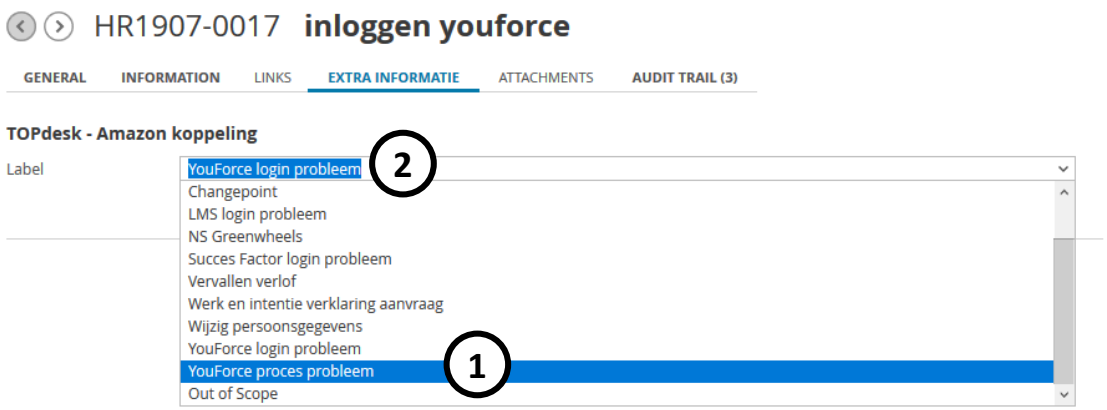


Figure 15: *The tab 'Extra Informatie', where employees can change labels to correct the AWS training data*



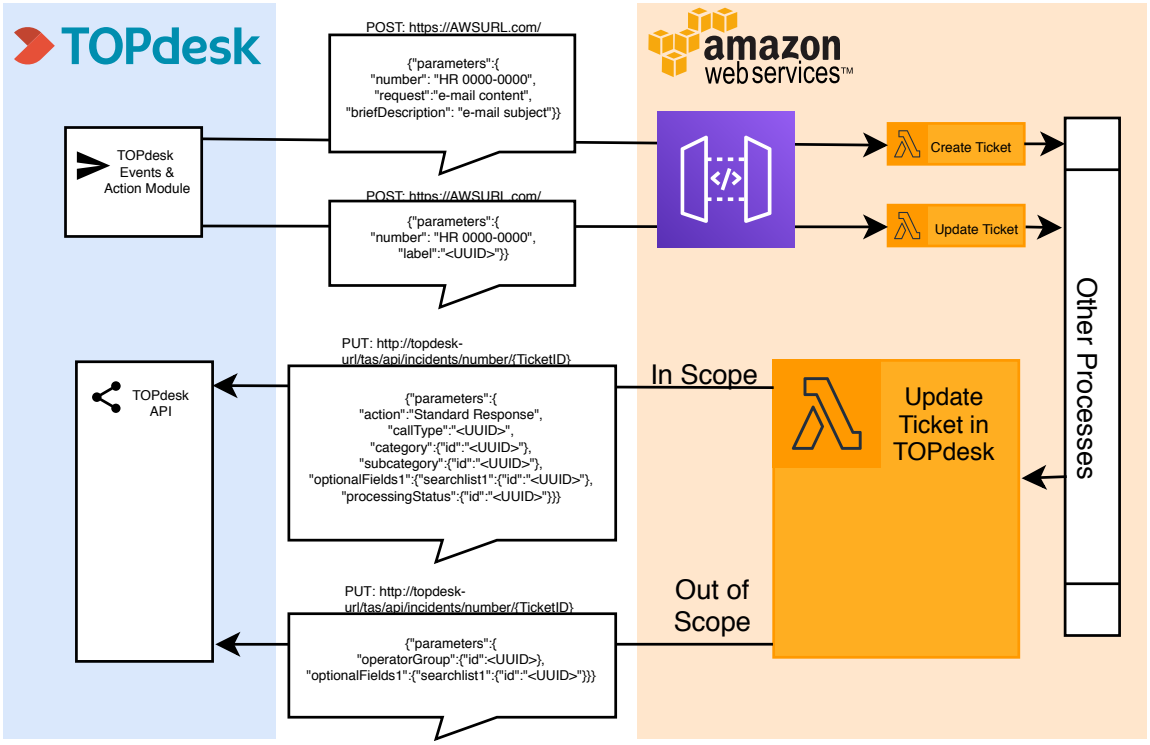
A.2 API PROTOCOL

Here, the API protocol between the two systems is described. Note that the 'header' element of a standard REST API request is left out, as no security details will be shown in the thesis. Furthermore, the term < UUID> stands for 'Unique Identifier'. These stand for a unique identifier for an element in TOPdesk. Communicating with these UUIDs ensures that if the visual name of something is changed, such as a label, the underlying reference, the UUID, remains the same. In this way, label names can be changed without consequences to the API, and label names can be added as well.

In figure 16, the API specification can be found. Here, *In Scope* means the eleven labels that are not 'NaN', and thus can be answered with a standard reply and *Out of Scope* refers to 'NaN', where a standard answer is not possible. Here, AWS changes the operator group.

Whilst the parameter names of the POST requests to AWS are straightforward, the one AWS parameter for the PUT requests to TOPdesk requires some elaboration. *optionalFields1, searchlist1* is the parameter where the label UUID will be entered. Other field names can be found in the TOPdesk API [34].

Figure 16: *API Protocol between TOPdesk and AWS.*



B Initial Dataset

Column Name	Type	No.Values	Description
Line	String	91464	'Firstline' or 'secondline (escalation)'
Personnel Number	Integer	69261	5 digit number
TicketID	String	91464	HR0000-0000
Name Sender	String	91463	<Last Name>, <First Name>
Organisation Sender	String	80527	Ziggo, Vodafone, etc.
Office Location Sender	String	90851	<Location><LocationID>
Ticket Type	String	90847	Incident, Request, Question
Category	String	90901	22 unique categories exist
Subcategory	String	90848	144 unique subcategories exist
Status	String	91464	'Signed off' or 'Processing'
Processor	String	91463	<Last Name>, <First Name>
Processing Group	String	91464	HelloHR, Payroll, etc.
Vendor	String	1170	<Vendor Name>
Target Date	Datetime	91464	YYYY-MM-DD HH:MM:SS
Received Date	Datetime	91464	YYYY-MM-DD HH:MM:SS
Solved Date	Datetime	90864	YYYY-MM-DD HH:MM:SS
Days Open	Integer	91464	$\mu \approx 6.3$, $\sigma \approx 15.3$, $Q_1 = 0$, $Q_2 = 2$, $Q_3 = 6$, max=438
Done	Bool	91464	True or False
Signed-off	Bool	91464	True or False
Execution	String	91464	'Signed off' or 'Processing'
Department	String	69327	<Business Unit><Location><Team><TeamID>
HR Business Partner	String	69315	<First Name><Last Name>(<Emp.Numb)>
Description	String	91464	Ticket Description (e-mail subject)
Request	String	91460	Flat e-mail text (not MIME)
Action	String	91023	Flat e-mail/ticket text (not MIME)
Attachments	String	89591	XML-format describing included images, files

C Details Text Cleaning

In this part of the appendix, a detailed description is given for the cleaning process of the text in the dataset, including code snippets, regular expressions and other tools used. All code was written for Python 3.7.2 [86].

C.1 Finding Headers

In order to filter headers from actions and tickets, the following regular expressions were used, of which matches were replaced with a 'header handle'. The TOPdesk header regular expression was also used to capture the name of the sender and processor, in order to remove these from the dataset.

E-mail Header pattern: captures all standard dutch and english e-mail headers

```
^(\bAntwoord.\baan|\bDatum|\bUrgentie|\bKopie|\bDate|\bAfzender|
\bDatum.\bverzonden|\bNaar|\bVan|\bVerzonden|\bAan|\bOnderwerp|
\bCC|\bFW|\bSent|\bFrom|\bTo|\bSubject):
\s*([^\n\r]*)?.+\n?){2,}',
options = IGNORECASE|MULTILINE
```

TOPdesk Header pattern: captures the TOPdesk header, present for all TOPdesk messages

```
^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]
(\d{4})\s(0[0-9]|1[0-9]|2[0-3]|[0-9]):([0-5][0-9])
\s([ÖA-z].*):\s',
options = MULTILINE
```

C.2 Finding openers, signatures and names

In order to find openers and signatures in our dataset, a script was written that would create two text files for used openers and signatures in the dataset. From these found signatures, regular expression patterns were created that replaced these with a opener or signature handle, respectively. Finally, using the names of the processor and sender of the ticket, names are replaced with a name handle.

Code used for finding openers and signatures. After finding the openers and signatures, another manual cleaning step was done to remove any mix up between openers and signatures.

```
import re

#This code is run on text for both Requests and Actions
#From the dictionary starters and signatures, only occurrences
#of starters >= 3 or occurrences of signatures >= 5 are written
#to file

openers = {}
signatures = {}
# Find all lines that start with a word and end with a comma
pattern= re.compile(r'^[A-Za-z\s].+,$', re.MULTILINE)
for idx, row in tqdm(df.iterrows(), total=len(df)):
    matches = re.findall(pattern, row[col])
    #if exactly 2 lines are found, a good chance exists
    #that these 2 lines are an opener and signature
    if len(matches) == 2:
        #Take only the first word of the opener if there are more
        if len(matches[0].split(' ')) > 1:
            opener = matches[0].split(' ')[0]
        else:
            opener = matches[0]
            opener = opener.replace(',', '').strip()
        # count the occurrence of openers
        if opener in openers:
            openers[opener] += 1
        else:
            openers[opener] = 1
        #Add the second value in full to signatures
        if matches[1].strip() in signatures:
            signatures[matches[1].strip()] += 1
        else:
            signatures[matches[1].strip()] = 1
```

Found openers in the dataset

	G-morgen	Goedendag	Groet	Hey	geachte
All	Geacht	Goedenmiddag	HI	Hi	goedemiddag
Allen	Geachte	Goedenmorgen	Ha	Hii	goedemorgen
Beste	Geachte/Dear	Goedmeiddag	Hai	Hoi	hallo
Besten	Gegroet	Goedmeorgen	Hallo	L.S.	heer/mevrouw
Collega	Goede	Goedmiddag	Halo	L.s.	hi
Collega's	Goedeavond	Goedmorgen	He	LS	hoi
Collegae	Goedemiddag	Goeiedag	Hee	Lieve	
Collega's	Goedemogen	Goeiemiddag	Heey	Ls	
Dag	Goedemorgen	Goeiemorgen	Hello	Morning	
Dear	Goeden	Good	HelloHR	Team	
Dears	Goedenavond	Goodmorning	HelloHr	beste	

Found signatures in the dataset

```
Alvast bedankt en groet,
Alvast bedankt en vriendelijke groet,
Alvast bedankt voor de hulp,
Alvast bedankt,
Alvast dank en groet,
Alvast dank,
Alvast hartelijk dank,
BR,
BVD,
Bedankt alvast,
Bedankt en groet,
Bedankt en groeten,
Bedankt,
Best Regards,
Best regards / Met vriendelijke groet,
Best regards / met vriendelijke groet,
Best regards,
Bij voorbaat dank,
Br,
Bvd,
Dank alvast,
Dank en groet,
Dank en groeten,
Dank je wel,
Dank je,
Dank vast,
Dank,
Dank, groet,
Dankje,
Dankjewel alvast,
Dankjewel,
Gr,
Gr.,
Greetings,
Groet en dank,
Groet,
Groet, Kind regards,
Groeten,
Groeten/ Regards,
Groetjes,
Grt,
Grt.,
Grtjs,
Grts,
Grtz,
Hartelijk dank,
Hartelijke Groet,
Hartelijke groet / Kind regards,
Hartelijke groet,
Hartelijke groet, kind regards,
Hartelijke groet/ Kind regards,
Hartelijke groeten,
Hoogachtend,
Hoor graag van jullie,
Hoor graag,
Hoor het graag,
Ik hoor graag van jullie,
Ik hoor het graag van jullie,
Ik hoor het graag,
KR,
Kind Regards / Met vriendelijke groeten,
Kind Regards,
Kind regards / Met vriendelijke groet,
Kind regards | Met vriendelijke groet,
Kind regards,
Kind regards, Met vriendelijke groet,
Kind regards, met vriendelijke groet,
Kind regards/Met vriendelijke groet,
Kind regards/Met vriendelijke groeten,
M.v.g,
M.v.g.,
MVG,
Many thanks,
Met Vriendelijke Groet / Kind Regards,
Met Vriendelijke Groet,
Met Vriendelijke groet,
Met dank en vriendelijke groet,
Met groet,
Met hartelijke groet / With kind regards,
Met hartelijke groet,
Met hartelijke groet/with kind regards,
Met vriendeijke groet,
Met vriendelijk groet / Kind regards,
Met vriendelijk groet,
Met vriendelijk groeten,
Met vriendelijke Groet,
Met vriendelijke groet ,
Met vriendelijke groet / Best regards,
Met vriendelijke groet / Kind Regards,
Met vriendelijke groet / Kind regards ,
Met vriendelijke groet / Kind regards,
Met vriendelijke groet / Regards,
Met vriendelijke groet / With kind regards,
Met vriendelijke groet / With regards,
Met vriendelijke groet / kind regards,
Met vriendelijke groet / with kind regards,
Met vriendelijke groet \ Kind regards,
Met vriendelijke groet | Kind Regards,
Met vriendelijke groet | Kind regards,
Met vriendelijke groet - Kind regards,
Met vriendelijke groet,
Met vriendelijke groet,  Kind regards,
Met vriendelijke groet, / With kind regards,
Met vriendelijke groet, Best Regards,
Met vriendelijke groet, Best regards,
Met vriendelijke groet, Kind Regards,
Met vriendelijke groet, Kind regards,
Met vriendelijke groet, With kind regards,
Met vriendelijke groet, kind regards,
Met vriendelijke groet, with kind regards,
Met vriendelijke groet,/ Best regards,
Met vriendelijke groet/ Best regards,
Met vriendelijke groet/ Kind regards,
Met vriendelijke groet/Best Regards,
Met vriendelijke groet/Kind Regards,
```

```
Met vriendelijke groet/Kind regards,
Met vriendelijke groet/With kind regards,
Met vriendelijke groet/kind regards,
Met vriendelijke groet/with kind regards,
Met vriendelijke groeten / Kind regards,
Met vriendelijke groeten / kind regards,
Met vriendelijke groeten,
Met vriendelijke groeten, kind regards,
Met vriendelijke groeten/ Kind regards,
Met vriendelijke groeten/ kind regards,
Met vriendelijke groeten/Kind regards,
MvG,
Mvg,
Mvgr,
Mvrgr,
Regards,
Rgds,
Rgs,
Sincerely,
Succes en vriendelijke groet,
Thank you in advance,
Thank you,
Thanks & Regards,
Thanks and Regards,
Thanks and best regards,
Thanks and regards,
Thanks in advance,
Thanks,
Thx,
Veel dank,
Verneem graag van jullie,
Vriendelijk bedankt,
Vriendelijk groet,
Vriendelijke Groet,
Vriendelijke groet / Kind regards,
Vriendelijke groet,
Vriendelijke groet/Kind regards,
Vriendelijke groeten,
Vriendelijke groeten/Kind regards,
With kind regards,
With regards,
gr,
groet,
groeten,
groetjes,
met vriendelijk groet,
met vriendelijke groet,
met vriendelijke groeten,
mvg,
vriendelijke groet,

Alvast bedankt en met vriendelijke groet,
Bij voorbaat dank en met vriendelijke groet,
Bij voorbaat mijn dank en met vriendelijke groet,
Met vriendelijke groet / Kind regards / Freundliche Grußen,
Met vriendelijke groet/ Kind regards/ Mit freundlichen grüßen/ Üdvözlettel,
Met vriendelijke groet / With friendly regards,
Met vriendelijke groet/ With kind regards/ Mit Freundliche Grüßen,
Met vriendelijke groet/ With Kind Regards,
Met vriendelijke groet/ With kind regards,
Met vriendelijke groeten, With kind regards,
Met vriendelijke groeten, with kind regards,
With kind regards / Met vriendelijke groet,
```

Code used to create a list of patterns from a list of standard openers and signatures of e-mails.

```
import re
import config # contains strings for global names

# First, create patterns for the openers
with open(f'{SRC_PATH}//{OPENERS_FILE}', 'r') as strt_file:
    opener_words = [word.strip() for word in strt_file.readlines()]
    open_patterns = [re.compile(r'^\b' + re.escape(word) + r'\b',\
    re.MULTILINE) \
    for word in opener_words]
# Second, create patterns for the signatures
with open(f'{SRC_PATH}//{SIGNATURES_FILE}', 'r') as sign_file:
    signature_words = [word.strip() for word in sign_file.readlines()]
    sig_patterns = [re.compile(r'^' + re.escape(word),\
    re.DOTALL|re.MULTILINE) \
    for word in signature_words]
# Then, replace the matching pattern with a handle for openers and signatures
# here, content represents the ticket content, either action or request
for pattern in open_patterns:
    content = re.sub(pattern, config.OPEN_HANDLE, content)
for pattern in sig_patterns:
    content = re.sub(pattern, config.SIG_HANDLE, content)
```

Code used to remove names.

```
import re
#Names are retrieved from the 5th capture group of the
#TOPdesk header regular expression.
#If these are names of employees, these are presented as
# 'Doe, John van'. We use John van Doe-Appleseed as an example.

name = name.lower()
#initialize a list with one value, 'Doe-Appleseed, John van'
names = [name]
if ',' in name:
    split_name = name.split(',')
    name = f"{split_name[1]} {split_name[0]}"
    name = name.strip()
    names.append(name) # 'John van Doe-Appleseed'
name = name.split(' ')
first_name = name[0].strip()
del name[0]
name = " ".join(name)
last_name = name.strip()
if '-' in name: #when people are married, multiple last names exist
    name = name.split('-')
    names.append(f"{first_name} {name[0].strip()}") # 'John van Doe'
    names.append(f"{first_name} {name[1].strip()}") # 'John Appleseed
else:
    names.append(f"{first_name} {last_name}") # 'John van Doe'
names.append(first_name) # 'John'
for name in names:
    pattern = re.compile(r'\b'+re.escape(name) + r'\b', re.IGNORECASE)
    content = re.sub(pattern, config.NAME_HANDLE, content) #ticket Action or Request
    desc = re.sub(pattern, config.NAME_HANDLE, desc) #ticket subject
```

C.3 Removing noise from requests and actions

Using the placed handles, the following regular expressions were used to remove everything that matched. Other handles, such as names and headers, were also removed.

Regular expression used to remove everything between and including signature handles and opener handles

```
fr'({config.SIG_HANDLE}.*?)(?:{config.HEADER_HANDLE})', re.DOTALL|re.MULTILINE)
```

Regular expression used to remove trailing signatures and everything afterwards

```
fr'{config.SIG_HANDLE}.*', re.DOTALL|re.MULTILINE)
```

Regular expression used to remove openers

```
fr'^(.+)?({config.OPEN_HANDLE}).*?(,|!|\.|\n)', re.MULTILINE
```

C.4 N-grams

N-grams can improve the quality of text by joining words that occur together frequently or a significant amount. For this project, Bigrams and Trigrams were collected, as well as their frequency and their PMI scores. To improve the overall quality of bigrams and trigrams, the text was tagged with Part of Speech tags using SpaCy [81], and only matches between Nouns and Adjectives were kept for Bigrams, and Nouns and Adjectives were allowed for w_1 and w_3 in the trigram, whilst everything was allowed for w_2.

The PMI scores were calculated with equations 8 and 9. However, the trigrams used for the evaluation of the effect of trigrams were not generated using the PMI equation below due to a mistake in the equation in the code, discovered too late. However the list of trigrams contains quite good examples, and from the perspective of the results, the impact of this mistake will not make a significant difference in the performance of the machine learning solution. The mistaken calculation used equation 10. Whilst bigrams and trigrams were created for both requests and actions, only request ngrams were used for evaluation. Ngrams are however not shown in this research due to confidential information present in some of these ngrams.

For a pair of words w_1 and w_2, the PMI of their bigram would be:

$$PMI(w_1; w_2) = log \frac{p(w_1, w_2)}{p(w_1)p(w_2)} \tag{8}$$

For a pair of words w_1 and w_2 and w_3, the PMI of their trigram would be:

$$PMI(w_1; w_2w_3) = log \frac{p(w_1, w_2w_3)}{p(w_1)p(w_2w_3)} \tag{9}$$

The mistake, however, used the equation below to calculate the PMI of a trigram:

$$PMI_{false}(w_1; w_2w_3) = log \frac{p(w_1, w_2w_3)}{p(w_1)p(w_2)p(w_3)} \tag{10}$$

Bigrams with a frequency higher than 100 or a PMI higher than 7 were kept, whereas trigrams with a frequency higher than 80 and a PMI higher than 20 were kept, using the wrong PMI function.

C.5 Tokenize

All tickets are tokenized using SpaCy [81], an NLP package for Python with a Dutch language model. Here, if the word is not a stopword, form of punctuation or if the part of speech is not anything else than a noun, adjective or verb, the lemma of the word is retained in the final content sent to the transformation that will encode the text into numerical features. Note that in the case of n-grams, n-grams will be added after this step.

Code used for tokenization using SpaCy [81]. This is the final cleaning step before transformation

```
import spacy

nlp = spacy.load('nl_core_news_sm', disable=['ner', 'textcat']) #disable unused models
content = ' '.join([t.lemma_.lower() for t in nlp(content)\ #lemmatize each word
if t.pos_ in ['NOUN', 'ADJ', 'VERB'] and \ # use only certain Part of Speech
t.text.lower() not in stopwords \ #remove stopwords
and t.text.isalpha()]) #remove punctuation
```

Dutch stopwords used for cleaning. Taken from the stopwords-iso project [87]

aan	daaruit	geleden	jouwe	nergens	terwijl	wat
aangaande	daarvanlangs	gelijk	juist	net	thans	we
aangezien	dan	gemoeten	jullie	noch	tien	wederom
achte	dat	gemogen	kan	nochtans	tiende	weer
achter	de	genoeg	klaar	nog	tijdens	weg
achterna	deden	geweest	kon	nogal	tja	wegens
af	deed	gewoon	konden	nu	toch	weinig
al	der	gewoonweg	krachtens	nv	toe	wel
aldaar	derde	haar	kun	of	toen	weldra
aldus	derhalve	haarzelf	kunnen	ofschoon	toenmaals	welk
alhoewel	dertig	had	kunt	om	toenmalig	welke
alias	deze	hadden	laatst	omdat	tot	werd
alle	dhr	hare	later	omhoog	totdat	werden
allebei	die	heb	liever	omlaag	tussen	werder
alleen	dikwijls	hebben	lijken	omstreeks	u	wezen
alles	dit	hebt	lijkt	omtrent	uit	whatever
als	doch	hedden	maak	omver	uitgezonderd	wie
alsnog	doe	heeft	maakt	ondanks	uw	wiens
altijd	doen	heel	maakte	onder	vaak	wier
altoos	doet	hem	maakten	ondertussen	vaakwat	wij
ander	door	hemzelf	maar	ongeveer	van	wijzelf
andere	doorgaand	hen	mag	ons	vanaf	wil
anders	drie	het	maken	onszelf	vandaan	wilden
anderszins	duizend	hetzelfde	me	onze	vanuit	willen
beetje	dus	hier	meer	onzeker	vanwege	word
behalve	echter	hierbeneden	meest	ooit	veel	worden
behoudens	een	hierboven	meestal	ook	veeleer	wordt
beide	eens	hierin	men	op	veertig	zal
beiden	eer	hierna	met	opnieuw	verder	ze
ben	eerdat	hierom	mevr	opzij	verscheidene	zei
beneden	eerder	hij	mezelf	over	verschillende	zeker
bent	eerlang	hijzelf	mij	overal	vervolgens	zelf
bepaald	eerst	hoe	mijn	overeind	via	zelfde
betreffende	eerste	hoewel	mijnent	overige	vol	zelfs
bij	eigen	honderd	mijner	overigens	volgend	zes
bijna	eigenlijk	hun	mijzelf	paar	volgens	zeven
bijv	elk	hunne	minder	pas	voor	zich
binnen	elke	ieder	misschien	per	vooraf	zichzelf
binnenin	en	iedere	mits	precies	vooral	zij
blijkbaar	enig	iedereen	mocht	recent	vooralsnog	zijn
blijken	enige	iemand	mochten	redelijk	voorbij	zijne
boven	enigszins	iets	moest	reeds	voordat	zijzelf
bovenal	enkel	ik	moesten	rond	voordezen	zo
bovendien	er	ikzelf	moet	rondom	voordien	zoals
bovengenoemd	erdoor	in	moeten	samen	voorheen	zodat
bovenstaand	erg	inderdaad	mogen	sedert	voorop	zodra
bovenvermeld	ergens	inmiddels	mr	sinds	voorts	zonder
buiten	etc	intussen	mrs	sindsdien	vooruit	zou
bv	etcetera	inzake	mw	slechts	vrij	zouden
daar	even	is	na	sommige	vroeg	zowat
daardoor	eveneens	ja	naar	spoedig	waar	zulk
daarheen	evenwel	je	nadat	steeds	waarom	zulke
daarin	gauw	jezelf	nam	tamelijk	waarschijnlijk	zullen
daarna	ge	jij	namelijk	te	wanneer	zult
daarnet	gedurende	jijzelf	neem	tegen	want	
daarom	gehad	jou	negen	tegenover	waren	
daarop	gekund	jouw	nemen	tenzij	was	

D Hyperparameters

This section contains the various hyperparameters used during the thesis, such as for the Latent Dirichlet Allocation and the experiments for each model. For the LDA, Gensim was used [82]. Moreover, different cleaning was done prior to the topic modelling, where the bigram and trigram tools of Gensim were used as opposed to the tools used in the rest of the thesis for n-gram extraction. Finally, words were tokenized and lemmatized using spacy, leaving only Nouns, Adjectives and Verbs as input for the topic model. In the case of the LDA, for some hyperparameters, multiple values were used. Here, multiple values are given.

Implementations from the scikit-learn package were used for the SVM [88]and RandomForest Classifiers [89]. Moreover, the SVM was wrapped using the OneVersusRestClassifier and OneVersusOneClassifier, also implemented through scikit-learn [90]. For XGBoost, a scikit-learn wrapper was used [91]. For the Bi-LSTM neural net, Keras was used with a TensorFlow backend [92]. When FastText or Word2Vec embedding weights were used to initialize the embedding layer of the Bi-LSTM neural net, this layer was made untrainable.

For the transforms, implementation from sklearn were used for the TF vectorization [93] and TF-IDF vectorization [94]. For Word2Vec, FastText and Doc2Vec, implementations from Gensim [82].

D.1 Hyperparameters Latent Dirichlet Allocation

Hyperparameter	Value
num_topics	5, 10, 20, 30
decay	0.5
random_state	100
update_every	1
passes	10
chunksize	30000
alpha	'auto'
per_word_topics	True

D.2 HYPERPARAMETERS TRANSFORMS

Transform	**Hyperparameter**	**Value**
CountVectorizer	-	-
TfidfVectorizer	max_df	0.3
	min_df	5
Word2Vec	size	100
	window	5
	min_count	1
	iter	10
FastText	size	100
	window	5
	min_count	1
	word_ngrams	3
	epochs	10
Doc2Vec	vector_size	100
	window	5
	min_count	1
	iter	10

D.3 HYPERPARAMETERS MODELS

Model	Hyperparameter	Value
One-versus-Rest SVM & One-versus-One SVM	C	100
	kernel	'rbf'
	gamma	'scale'
	class_weight	'balanced'
	decision_function_shape	'ovr' & 'ovo'
RandomForestClassifier (untuned)	n_estimators	400
	min_samples_split	10
	min_samples_leaf	4
	max_features	'sqrt'
	max_depth	'None'
	bootstrap	False
	class_weight	'balanced'
RandomForestClassifier Tuned with Unigrams, FastText	n_estimators	400
	min_samples_split	5
	min_samples_leaf	2
	max_features	'sqrt'
	max_depth	50
	bootstrap	False
	class_weight	'balanced'
XGBoostClassifier	learning_rate	0.025
	n_estimators	400
	objective	'multi:softmax'
	num_class	12
	max_depth	20
	eval_metric	'mlogloss'
	early_stopping_round	10
	eval_set	(test_docs, test_labels)
	sample_weight	compute_sample_weight('balanced')*
	sample_weight_eval_set	compute_sample_weight('balanced')*
Bi-LSTM	Embedding	(None, 250, 100), params = Vocab_size × 100
	Bidirectional LSTM	(None, 200), dropout=0.2, params = 160800
	Dense	(None, 12), params = 2412
	loss	'categorical_crossentropy'
	optimizer	'adam'
	metrics	['accuracy']
	epochs	10
	batch_size	64
	sample_weight	compute_sample_weight('balanced')*
	validation_data	(test_docs, test_labels)

**sample_weights were calculated using the class ratio and applied to either the index of training or test labels in a fold*

E Detailed Results

In this section, detailed results are presented for the most optimal configurations of data, transformer and classification model as determined in section 7.1 'Experiment 1'. Furthermore, the hyperparameters of the RandomForest classifier have been further tuned, using the combination of unigrams and FastText.

The SVM model was already optimized during the label expansion. The XGBoost and Bi-LSTM models were not optimized due to a lack of computing power to do this in a considerate amount of time.

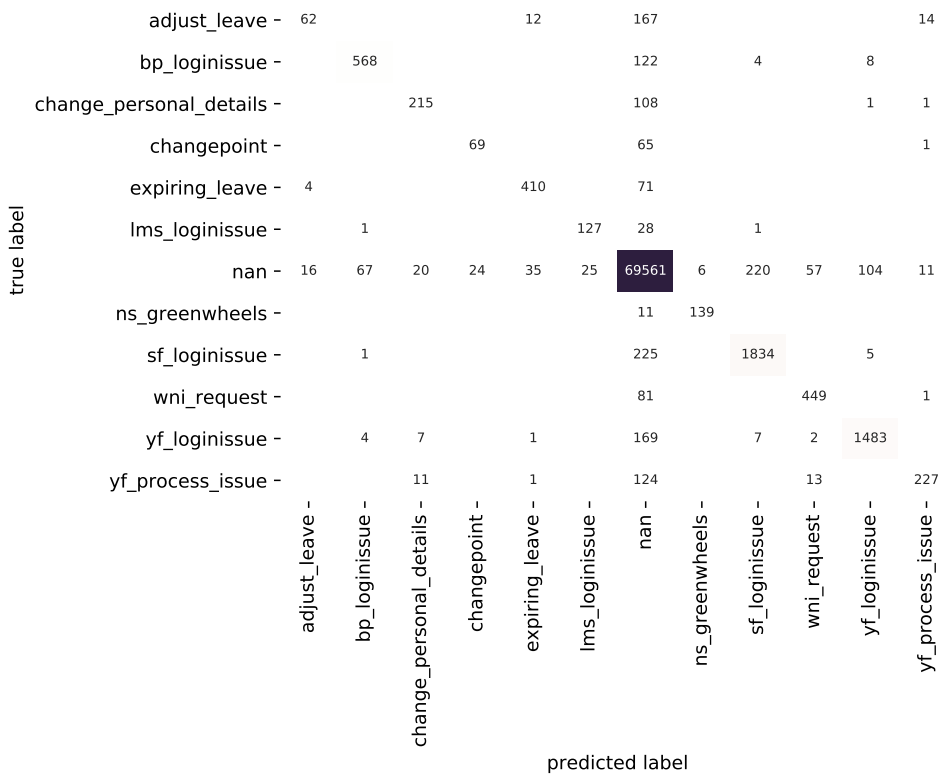
$F_{0.5}$ scores, precision scores or recall scores that are boldfaced are the best scores across models. The scores presented are the mean of a shuffled stratified 5-fold cross-validation. The confidence interval is given for 95% confidence. Moreover, for each configuration, a confusion matrix is provided.

To summarize these detailed results: the SVM is the best choice for a precise system that scores the best based on the $F_{0.5}$ evaluation criteria and therefore has the best mix of precision and impact. If one would desire a system with more impact, tolerating low precision, the Bi-LSTM, whilst very basic, already shows high potential with an estimated 7.7% higher impact on a micro-level.

E.1 Unigram + TF-IDF + OVR + SVM

Class label	F0.5	CI +/-	Precision	CI +/-	Recall	CI +/-
adjust_leave	**0.529**	0.209	0.753	0.283	0.243	0.113
bp_loginissue	**0.869**	0.020	**0.886**	0.022	0.809	0.046
change_personal_details	**0.804**	0.041	**0.851**	0.053	0.662	0.050
changepoint	**0.682**	0.136	**0.750**	0.174	0.511	0.155
expiring_leave	**0.884**	0.082	**0.895**	0.096	0.845	0.065
lms_loginissue	**0.830**	0.097	0.836	0.101	0.809	0.120
ns_greenwheels	**0.952**	0.059	0.959	0.064	0.927	0.049
sf_loginissue	**0.888**	0.025	**0.888**	0.038	0.888	0.030
wni_request	**0.858**	0.037	**0.862**	0.035	0.846	0.079
yf_loginissue	**0.918**	0.018	**0.927**	0.030	0.886	0.046
yf_process_issue	**0.814**	0.042	0.892	0.070	0.604	0.032
Macro	**0.821**	0.040	**0.864**	0.054	0.730	0.026
Micro	**0.874**	0.016	**0.891**	0.019	0.815	0.008

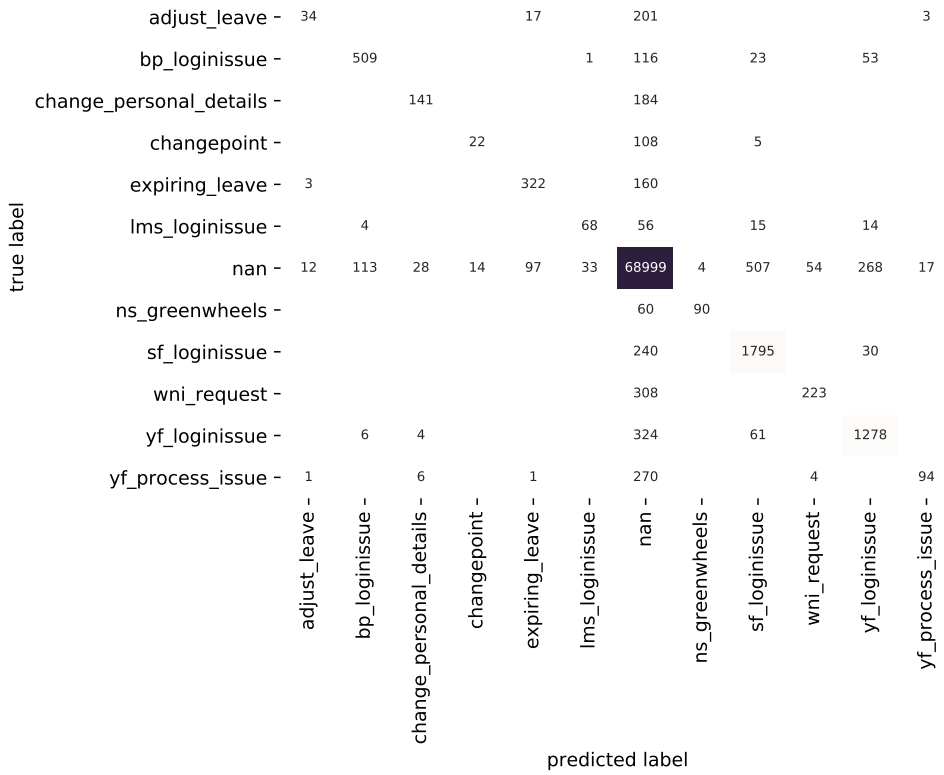
Figure 17: *Confusion Matrix SVM.*



E.2 Unigram + FastText + RandomForest

Class label	F0.5	CI +/-	Precision	CI +/-	Recall	CI +/-
adjust_leave	0.159	0.122	**0.820**	0.437	0.039	0.034
bp_loginissue	0.788	0.030	0.854	0.032	0.601	0.029
change_personal_details	0.676	0.054	0.848	0.058	0.375	0.068
changepoint	0.267	0.297	0.660	0.453	0.096	0.149
expiring_leave	0.739	0.100	0.839	0.105	0.503	0.107
lms_loginissue	0.776	0.124	**0.888**	0.042	0.528	0.227
ns_greenwheels	0.871	0.059	**1.000**	0.000	0.580	0.135
sf_loginissue	0.790	0.017	0.798	0.013	0.761	0.046
wni_request	0.652	0.052	0.824	0.048	0.358	0.072
yf_loginissue	0.816	0.035	0.867	0.043	0.660	0.042
yf_process_issue	0.451	0.153	**0.946**	0.092	0.152	0.078
Macro	0.635	0.057	0.849	0.066	0.423	0.044
Micro	0.764	0.014	0.835	0.015	0.570	0.023

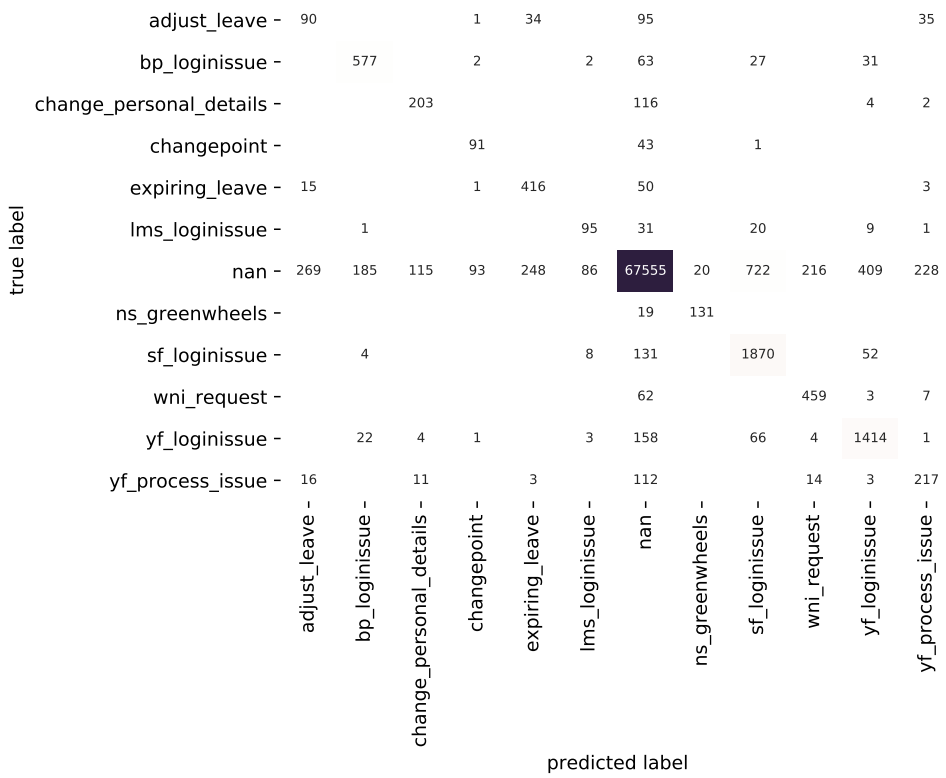
Figure 18: *Confusion Matrix RandomForest.*



E.3 Unigram + Word2Vec + XGBoost

Class label	F0.5	CI +/-	Precision	CI +/-	Recall	CI +/-
adjust_leave	0.249	0.108	0.232	0.102	0.353	0.156
bp_loginissue	0.748	0.050	0.732	0.052	0.822	0.038
change_personal_details	0.613	0.088	0.611	0.093	0.625	0.089
changepoint	0.515	0.076	0.488	0.089	0.674	0.125
expiring_leave	0.634	0.090	0.596	0.094	0.858	0.074
lms_loginissue	0.514	0.081	0.499	0.107	0.604	0.103
ns_greenwheels	0.872	0.155	0.872	0.172	0.873	0.096
sf_loginissue	0.726	0.030	0.691	0.033	0.906	0.018
wni_request	0.695	0.053	0.663	0.053	0.864	0.056
yf_loginissue	0.754	0.043	0.735	0.045	0.845	0.044
yf_process_issue	0.462	0.076	0.440	0.076	0.577	0.085
Macro	0.617	0.019	0.596	0.019	0.727	0.024
Micro	0.677	0.018	0.650	0.020	0.812	0.019

Figure 19: *Confusion Matrix XGBoost.*



E.4 Unigram + No weights + Bi-LSTM

Class label	F0.5	CI +/-	Precision	CI +/-	Recall	CI +/-
adjust_leave	0.244	0.079	0.214	0.080	**0.612**	0.198
bp_loginissue	0.732	0.089	0.701	0.107	**0.897**	0.072
change_personal_details	0.573	0.133	0.540	0.150	**0.785**	0.129
changepoint	0.552	0.113	0.512	0.126	**0.830**	0.109
expiring_leave	0.680	0.074	0.645	0.079	**0.874**	0.134
lms_loginissue	0.570	0.122	0.524	0.136	**0.904**	0.106
ns_greenwheels	0.895	0.100	0.883	0.125	**0.953**	0.067
sf_loginissue	0.791	0.078	0.759	0.090	**0.951**	0.034
wni_request	0.736	0.023	0.703	0.028	**0.908**	0.046
yf_loginissue	0.772	0.074	0.742	0.087	**0.922**	0.033
yf_process_issue	0.495	0.132	0.463	0.135	**0.702**	0.160
Macro	0.640	0.058	0.608	0.067	**0.849**	0.036
Micro	0.687	0.074	0.651	0.085	**0.892**	0.020

Figure 20: *Confusion Matrix Bi-LSTM.*