Artificial Intelligence Appliances in Customer Communications Management solutions

Juuso Heljaste

Haaga-Helia University of Applied Sciences
Business Technologies
Information Services and Systems
Thesis
2023

Abstract

Author(s)
Juuso Heljaste

Degree
Master of Business Administration

Thesis title
Artificial Intelligence Appliances in Customer Communications Management solutions

Number of pages and appendix pages
41 + 3

The main objective of this constructive research study was to find suitable use cases of Artificial Intelligence (AI) appliances in Customer Communication Management (CCM) business domain. The second main objective was to create a prototype for the most feasible identified use case while describing best practices guidelines on how to implement such modules utilizing open-source tools. CCM domain is still in its early phases of adoption of AI features. Although many AI branches seem prominent to CCM, this study was limited to Machine Learning (ML) and Natural Language Processing (NLP) AI branches.

The theoretical framework part of the study started with an overview of AI, its history, and description of different AI branches. AI section was followed by ML overview and practical implementation guidelines for ML systems and then by similar structure with NLP. Theoretical framework was concluded with an overview of CCM domain and description of typical business cases identified by the commissioner of the study, ENIT AB.

The main data collection method was a brainstorming session held among ENIT subject matter experts in March 2023. Each topic identified and discussed during the brainstorming session was analysed using applicable content analysis methods and evaluated within the concept of ML and NLP. Feasibility as an AI use case for ENIT and CCM domain in general was also analysed. The outcome of the analysis was an evaluation table of each topic in terms of complexity of implementation and possible value to CCM. The use case identified as the low-hanging-fruit was selected as the foundation for constructing the prototype.

For the prototype, a Machine Learning classifier for important log entries was successfully developed. The implementation phases were described in detail providing a solid framework for ENIT and other CCM providers to create similar features in their product and service offering. The concept of the prototype classifier is applicable throughout CCM industry.

The planning of the study started in January 2023 and the study was concluded in November 2023. The main challenges encountered revolve around the scarcity of scientific research combining AI and CCM. However, many feasible AI use cases in CCM were able to be identified and many of them can be developed with moderate effort. Further research on other AI branches should follow this study to fully unlock AI's potential in CCM outside of ML and NLP branches.

Keywords
CCM, AI, NLP, ML

Table of contents

1 Introduction

Artificial Intelligence (AI) and Machine Learning (ML) have been a profound interest for me for the several years now. I have been studying them via LinkedIn and similar online learning platforms before my master studies in Haaga-Helia, especially ML. During my master studies I have enrolled to every possible course regarding AI, robotic process automation and ML. I have done some experimental machine learning and natural language processing (NLP) modelling myself but none of the models are in any actual production use.

I work as a Senior Consultant in ENIT, a company providing customer communication management (CCM) solutions, software, consultancy, and service. My role is a combination of pre-sales engineer, solution architect and software engineering. Having worked in CCM business more than 10 years, I would summarize CCM being everything around managing and operating customer communication solutions. This includes all outgoing communication such as emails, SMS notifications, paper letters and documents and their corresponding digital formats. CCM is not just about the delivery of documents and their contents but also archiving and change management for the contents of those documents and notifications, while not forgetting integrations and third-party dependencies. This is quite aligned with what that Gartner (2023) defines as CCM saying it is a strategy focusing on improving creation, delivery, storage, and retrieval of different outbound communications like marketing, product information and manuals and various notifications around customer events. CCM is rather new acronym and perhaps the term “output management” is more familiar, but in this study, I shall use the term CCM.

The commissioner of this research is ENIT AB, which also serves as my employer. ENIT has been founded in late 2015 and currently has 27 employees located in Norway, Sweden, and Finland and has few subcontractors as well. ENIT is the leading independent CCM and output management specialized provider and its focus in on document creation and design, orchestration, and distribution. We have combined working experience of over 400 years in CCM and other relevant areas. One of the reasons ENIT is acting as a commissioner of this study is the fact, that ENIT is still in its very early stages of implementing AI into their solutions and products. This study also aims to provide ENIT a way forward on its path with AI.

My main research questions for the study are:

- What are the key areas within CCM where AI can be effectively utilized?
- What are the best practices and strategies for successfully implementing AI applications in the context of CCM?

I consider this study's topic urgently important, mostly as I think not only ENIT, but CCM providers in general have a continuously growing technical debt when it comes to fully utilizing AI. To my best understanding there are not many, if any, scientific research studies combining AI and CCM. CCM is in my opinion already falling behind with AI compared to other business domains and I would like that be changed. There are also plenty of old legacy systems widely in use where AI could be used to help migrating away from. This has been recognized also by Nelms and Phifer (2021) who indicate the importance of applying AI to CCM solutions for helping rationalizing legacy CCM systems and old implementations among other software development areas.

Our company's main expertise is based on another company's product called Exstream (formerly StreamServe) developed and maintained by OpenText. Exstream has been recognized as a "Leader" both by Gartner in its magic quadrant for CCM (Gartner 2017), and by Aspire Leaderboard (Aspire 2023) in its leaderboard. ENIT also provides support for other CCM platforms, and we have built our own solution as well. In addition to supporting multiple CCM solutions ENIT has a wide offering of addons to CCM solutions to improve the overall quality and control of the CCM solution. However, none of these addons have ML or NLP features in them. While we have a steadily growing customer base, our products and services are not in any analyses by Gartner and Aspire or similar institutions as ENIT is not sponsoring neither of those at the time of writing.

Furthermore, from what we have observed in our company, it seems that CCM is on its way to its second hype phase. The first hype for CCM occurred around year 2010 and CCM was considered one of the top priorities in many companies back then. The interest slowly diminished since, but from what we hear from our customers, it has been taken up lately again, mostly in the form of instant and omnichannel customer experience expectations. Unfortunately for CCM in general, as the interest in CCM diminished it seemed that R&D efforts in various CCM companies. This has led to the situation where AI are not included in the top CCM solutions as much as they are included in many other business areas. Luckily, since the launch of ChatGPT in late 2022, the world has "awaken" to the possibilities of AI and OpenText among others has launch their own AI initiative (OpenText 2023).

Thormundsson (2023) estimates AI market size to grow up to four times its size, from 428 billion USD to 2025 billion USD, by the year 2030, while Research and Markets (2023) estimate CCM business to roughly double its market size, from 1.9 billion USD to 4.6 billion USD, by the year 2030. Since AI is growing with such great pace, I believe there is readily achievable potential to apply its features and general interested towards AI to CCM and help CCM grow alongside with AI. As both CCM and AI market sizes are growing, consumers and end-users must be growing as well. Increased market sizes will also mean more and more data which leads to enormous

potentials gained through learning from the data. According to Taylor (2023) there is approximately 330 million terabytes of data being created each day now and that amount will continue to grow, and according to Nelms and Phifer (2021) digital communication will continue to expand but old legacy implementation and systems might hinder that progress. Applying AI for learning from existing data and having AI features to help with the implementations should help with rationalization and modernization those systems.

1.1 Scope and limitations

As part of the study an AI prototype module will be developed. The AI use case for the prototype creation will be selected based on information gathered in the theoretical framework part of the study and combining that to potential use cases identified during the data collection phase.

ENIT has selected Node.js as its main programming platform, so the prototype will be utilizing that platform. Python is widely used programming language for AI and most parts of the prototype will be using Python. While many cloud service providers offer their own set of AI tools (with added cost mostly), this prototype will be implemented using open-source tools, libraries, algorithms, and other modules which allow commercial use. No separate budget has been allocated to this study, therefore experimenting with ready-made cloud service provider AI tools will be left to another study.

The prototype, as later described more in detail, will be using ML techniques. In addition to ML, NLP AI branch will be described more in detail as those two technologies seem most prominent for CCM according to Aspire (2023). There are obviously more AI branches that CCM could benefit from, especially deep learning technologies show great potential, but this study will be limited to short overview of AI and then focusing ML and NLP branches. Apart from the prototype implementation chapter, the study is otherwise aimed to be kept at rather high business information level without going too deep into technical details, although in certain parts of the study it cannot be avoided.

The study will be limited to observations based in Europe, and even more specific in the Scandinavia area as that is where ENIT mainly operates and where I have mostly worked. Certain of the references might include insights from the US and other parts of the Europe, but mostly this study is done from Scandinavian viewpoint.

1.2 Study structure

The study starts with reviewing what existing literature and publications state about artificial intelligence and its ML and NLP branches. AI, ML and NLP overview is followed by a brief overview of

CCM which will conclude then theoretical framework part of the study and ending chapter 2. Chapter 3 is about data collection and applying the knowledge gathered in chapter 2 for identifying the most prominent use case for the study's prototype creation. Chapter 4 describes the development process of the chosen prototype AI feature and explains the different phases of the implementation process in detail. The last chapter summarizes the findings of the study and concludes the study.

2 Overview of artificial intelligence

This chapter consists of four subchapters. The first subchapter describes AI in general, its history and its different branches. The following two subchapters describe ML and NLP AI technology branches more in detail, both of which are widely adopted in many difference business areas, and more specifically because they seem most prominent for CCM (Aspire 2023). Part of these technologies are also used in building the prototype for ENIT as describer later in the study.

The last sub chapter describes CCM more in detail for helping to understand what it is about. The CCM domain in that subchapter is described mostly from ENIT perspective.

2.1 Artificial intelligence

Artificial intelligence is about making computers understand human intelligence (McCarthy 2007, 2). McKinsey's consultants (2017) define AI as enabling machines to exhibit human-like cognition. Everdeen (2017) defines AI being machines performing intelligent activities. I like these definitions as they are quite well aligned with my definition of AI. For me AI is about computers being able to perform tasks that typically require human-like observation capabilities, reasoning, and decision-making.

AI is not a new thing at all. McCarthy, the father of AI as some call him, came up with the term Artificial Intelligence in 1955 (Manning 2020). In the year 1956 McCarthy organized a conference called Dartmouth Summer Research Project on Artificial Intelligence and that has been considered as the founding event of AI field (Dartmouth 2023). However, I like to think that AI got birth when Alan Turing and his team created the Bombe machine to help cracking the Enigma Machine, enciphering machine, during the second world war. After all, the Bombe made human-intelligent like decision making in this process (Elisbury 1998) and that falls into the categories of AI definition described above.

AI continued its development until mid-1970's while computing power and storage started to become more accessible and cheaper. Machine learning algorithms got improved and the users learned to select the appropriate algorithms better. However, the computation power and the amount of information that computers could store during that period were not fulfilling the needs the AI computation would have required to be efficient and fast enough. At this point, the expectations on AI fell short and development mostly halted for almost ten years. (Anyoha 28 August 2017). The period between mid-1970's to early 1980's was also called AI winter (Yang 17, 2006).

In the 1980's AI started to receive more and more funding. These resulted in development of new AI technologies such as deep learning and expert systems. By the start of the 21st century, the

computation power and storage capacity demands were starting to get resolved as they become increasingly accessible and considerably cheaper. This removed the earlier restraints on lack of storage capacity and with continuously increasing computing speed, AI could now solve the problems it was given quick enough and therefore enabling AI to continue its evolvement and growth. (Anyoha 28 August 2017).

Artificial Intelligence has many different branches but there are different opinions on what the branches should be. Davenport, Brynjolfsson, McAfee, and Wilson (2019) separate AI into only to two branches which are cognition and perception. By cognition Davenport et al (2019) refer to problem solving and learning, and by perception to voice recognition and visual observations both for motion and still pictures. McCarthy (2007) himself identifies quite many more AI branches. He considers them being logical AI, search, pattern recognition, representation, inference, common sense knowledge and reasoning, learning from experience, planning, epistemology, ontology, heuristics, and finally genetic programming. However, according to Blochi (2023) AI has only six branches which are machine learning, neural networks, expert systems, robotics, natural language processing, and fuzzy logic. When asked from AI itself, ChatGPT in this case, the categorization is yet a bit different. In addition to the ones already mentioned, ChatGPT also mentions AI ethics as one of the branches.

Even though there are no unified consensus for the branches, there seems to be common themes around them. To summarize, we can quite safely define common AI branches to be categorized in the following way:

- Machine learning and its subfields, deep learning, and neural networks.
- Nature language processing
- Computer vision and speech recognition
- Generative AI
- Robotics

For ENIT, all other than robotics can be considered a branch of interest, and even robotics can be included, if robotic process automation (RPA) is considered as part of it. But here I refer to robotics as performing actual physical actions, like a manufacturing site, and since ENIT is not operating in that area, it can be considered out of interest for ENIT. ML and NLP will be described more in detail in the following chapters as they have been already highlighted of high importance and potential for CCM and therefore for ENIT. Computer vision and speech recognition and Generative AI branches seem very prominent, but studying their possibilities for ENIT and CCM will be left for another study.

2.2 Machine Learning

This subchapter describes what is machine learning (ML) in rather high level. Machine learning is considered as one of the main branches of artificial intelligence as described in the previous subchapter. This chapter starts with on overview of ML and then proceeds on research on practical implementation of ML systems.

2.2.1 ML overview

ML is all about letting the machine (i.e., the computer in this case) to make an action or propose an action to a problem and it if provided feedback should learn better after each decision (Mitchell, 1997). Some easily understandable examples of applied ML are classifying an incoming email as spam or non-spam or to recommend something to watch on Netflix. ML application will make that classification or recommendation and the user can decide what to do with that information.

A continuously learning ML model will adjust its parameters according to the user actions and will therefore “learn” and make the next decision with higher probability of correctness. Sandramouli and Das (2018, chapter 1) state that machine can learn when it can apply its experience from one task and use that to improve its actions for similar upcoming tasks where Esposito and Esposito (2020, chapter 2) define machines learning as use of statistical methods for creating a model which can classify existing data or predict future events. Both authors and I share the same understanding that machine should learn from its past actions and improve its performance with similar activities in the future.

Sandramouli and Das (2018, chapter 1) divide machine learning in three broad categories:

- supervised learning
- unsupervised learning
- reinforcement learning

So does Rouhiainen (2020, chapter 1) as illustrated in below figure 1.

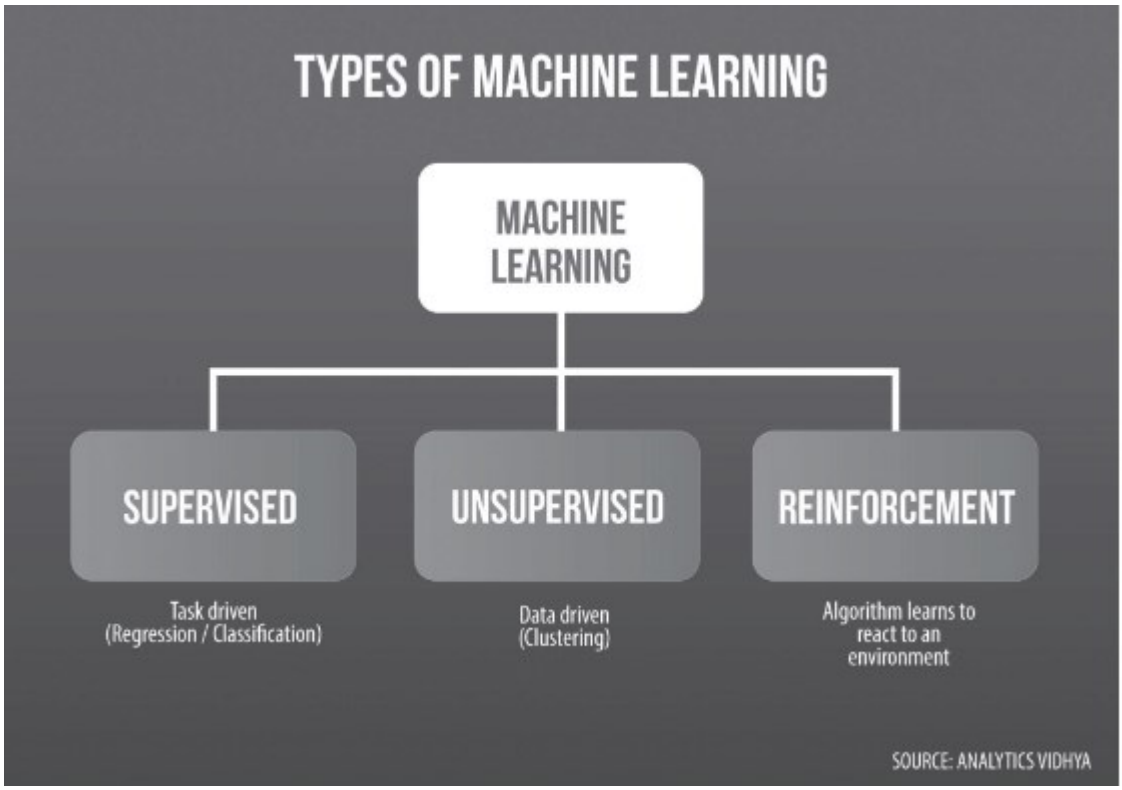


Figure 1. Types of Machine Learning (Rouhiainen 2020)

In supervised learning the machine uses classified data samples from the past to classify new unclassified data. For the email-spam-classifier example case, the ML application should classify whether the new incoming email is spam based on prior emails which have been labelled as spam or non-spam. One of the ways to call this prior data is *training data*. In this training phase, a prediction model is created which then used for the classification process to predict new data. (Sandramouli & Das 2018, chapter 1). In the below figure 2, there are sample applications for supervised learning systems to further illustrate what it can be used for.

Supervised learning systems

As two pioneers in the field, Tom Mitchell and Michael I. Jordan, have noted, most of the recent progress in machine learning involves mapping from a set of inputs to a set of outputs. Some examples:

Input X	Output Y	Application
Voice recording	Transcript	Speech recognition
Historical market data	Future market data	Trading bots
Photograph	Caption	Image tagging
Drug chemical properties	Treatment efficacy	Pharma R&D
Store transaction details	Is the transaction fraudulent?	Fraud detection
Recipe ingredients	Customer reviews	Food recommendations
Purchase histories	Future purchase behavior	Customer retention
Car locations and speed	Traffic flow	Traffic lights
Faces	Names	Face recognition

Figure 2. . Supervised Learning systems (Mitchell et al 2019)

In unsupervised learning there is no pre-classified data available for the machine to learn from or to predict from. Unlike in supervised learning the objective for unsupervised learning is not to classify or to predict on outcome, but rather finding similarities, natural groupings, or patterns from the given dataset. Grouping similar objects together in group and dissimilar objects to another is called clustering which is considered main type of unsupervised learning. Another main variant of unsupervised learning is association analysis where association between data entries is identified. (Sandramouli & Das 2018, chapter 1). The below figure 3 is an oversimplified application of unsupervised learning model, but in all its simplicity it underlines unsupervised learning's main purpose perfectly for me.

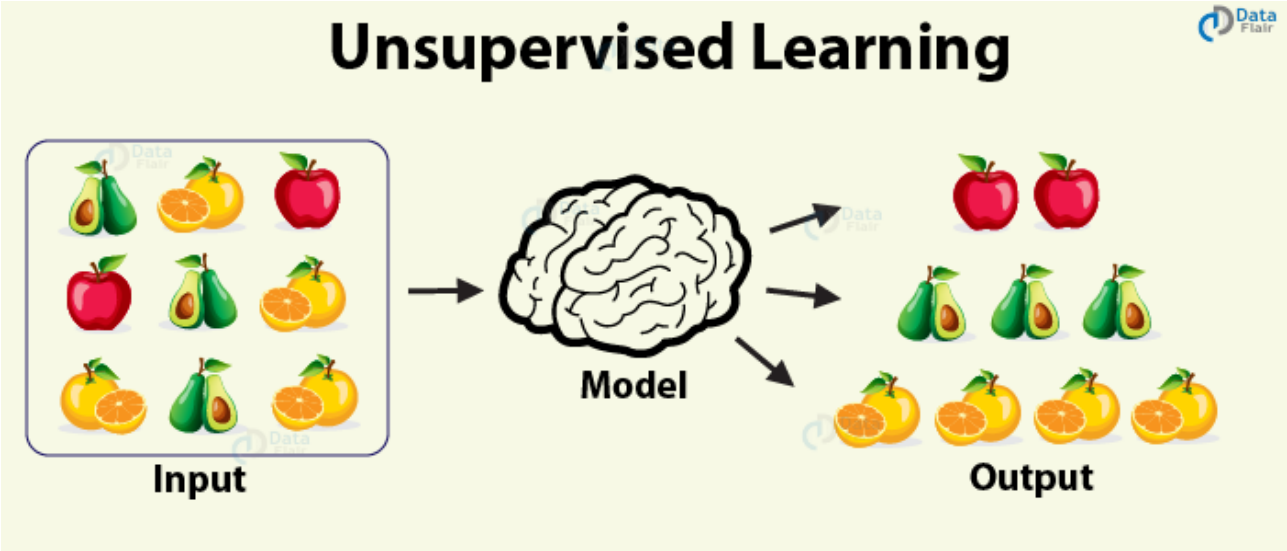


Figure 3. Unsupervised Learning (DataFlair 2023)

Reinforcement learning is mainly used to make predictions and classifications as in supervised learning main objectives, but unlike in supervised learning, reinforcement learning model does not get pre-labelled data to learn from. Instead, reinforcement learning application is given unclassified raw input data. The model classifies the data based on patterns and similarities and then learns by a human telling whether the ML application's classification was right or wrong. Human verifying model's correct answers indicates that the ML application is on the correct path, and vice versa for incorrect answers. (Sandramouli & Das 2018, chapter 1).

2.2.2 ML application development

As described in the introduction, I have some experience in building ML applications. Gollapudi and Laxmikanth (2016, chapter 1) describe developing a ML application as three-phased process. The first phase is preparing the training data and create the initial ML model. Initial model creation consists of selecting the best-suitable algorithms for the given task and adjusting its parameters. Following the initial model creation is the testing phase of the model with some test data. Many of the ML algorithms come with scoring and accuracy functions which indicate how accurate the trained model is. Depending on these scores and indicators, some adjustments might be needed for the model's parameters. Via an iterative process of finetuning the model and testing it again, we end up with the best possible model (for the time being) which we can then apply to the actual real-world data as a prediction model to be called from the actual application used to classify the data. (Gollapudi & Laxmikanth 2016, chapter 1).

To summarize Gollapudi and Laxmikanth (2016, chapter 1) describe this process in three phases. They call the phases in following names:

- Training Phase
- Validation and Test Phase.
- Application Phase.

In training phase, the model is trained by pairing the input data with expected output. In second phase the model is evaluated and its properties like error measures, recall and precision are estimated and evaluated. In the last phase the model is applied to real-world data and taken into use in a production environment. (Gollapudi & Laxmikanth 2016, chapter 1).

The below figure 4 by Sandramouli and Das (2018) illustrate the simplified process of supervised learning which can be applied with little changes to many machine learning models.

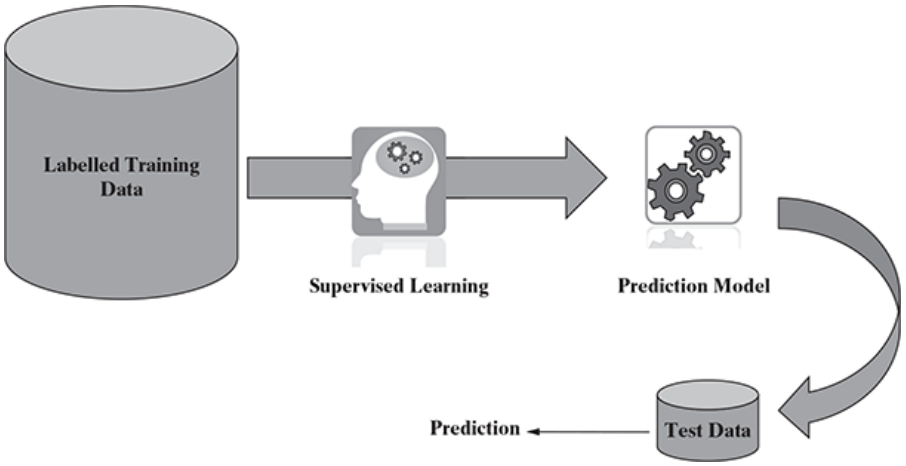


Figure 4. Supervised learning. (Sandramouli & Das 2018, chapter 1)

Most, if not all, internet users can be considered as consumers of machine learning applications (Gollapudi & Laxmikanth 2016, chapter 1) so it is fair to say it is a widely used technology. Typical use cases in modern society are facial recognition used in social media and on modern smartphones, speech recognition, product recommendations, stock trading and many more (Gollapudi & Laxmikanth 2016, chapter 1).

Out of these use cases, speech recognition is closely linked to CCM domain and could be an intriguing topic for ENIT to be evaluated in their AI journey later. Classifications also seem viable use case for ENIT as does the unsupervised learning possibilities to learn from the data. All in all, ML shows great potential for ENIT to start its AI journey.

2.3 Natural language processing

Natural language processing (NLP) is another one of the main branches of AI. Many NLP systems use ML techniques to understand and process human language. (Gupta, Majumber, Surana & Vaijala 2020, Preface). As with the ML subchapter, the first subchapter of this chapter describes NLP in high level, and the second subchapter describes the more practical NLP software development processes and practices. Throughout the chapter there is an aim to identify intriguing use cases for ENIT on its AI journey.

2.3.1 NLP overview

NLP influences our world probably more than most of realize. NLP models are in use in Google search engines, chatbots, social media platforms, and most smartphones have NLP build into them. Social media posts can be done by an NLP model instead of a natural person and even entire movie scripts can be done by NLP systems. NLP is used everywhere nowadays and its role in

our current modern society cannot be underestimated. Autocomplete and autocorrect functions are built with NLP models and have become normal features of modern human interactions with computers. (Hapke, Howard & Lane 2019, Front Matter). Some of the other use cases and their classifications as described by Hapke et al (2019) can be seen in the below figure 5.

Search	Web	Documents	Autocomplete
Editing	Spelling	Grammar	Style
Dialog	Chatbot	Assistant	Scheduling
Writing	Index	Concordance	Table of contents
Email	Spam filter	Classification	Prioritization
Text mining	Summarization	Knowledge extraction	Medical diagnoses
Law	Legal inference	Precedent search	Subpoena classification
News	Event detection	Fact checking	Headline composition
Attribution	Plagiarism detection	Literary forensics	Style coaching
Sentiment analysis	Community morale monitoring	Product review triage	Customer care
Behavior prediction	Finance	Election forecasting	Marketing
Creative writing	Movie scripts	Poetry	Song lyrics

Figure 5. Categorized NLP applications (Hapke et al 2019, chapter 1)

Hapke et al (2019, chapter 1) define NLP being about making computers understand human language. Gupta et al (2020, Preface) are quite aligned with that definitions as they define it being about building systems which understand and can process human language. A word for a computer is simply a set of characters in binary format. Therefore, by default, a computer does not understand the concept or a meaning of a word, as they are just numbers to it. This is where NLP comes into play, helping the computer to understand the meaning of a set of characters, words, and eventually set of words making up a sentence. Typical NLP model transforms words into numbers and uses statistical techniques and different algorithms to classify words and their typical contexts and meanings. (Hapke et al 2019, chapter 1).

NLP can be used to classify texts (Gupta et al 2020, Preface) which in my opinion seem very prominent to CCM. Text classification can be helpful when analysing for example customer messages and the tonality of outgoing customer communication. Both Gupta et al (2020, chapter 1) and Hapke et al (2019, chapter 1) are using chatbots as sample NLP systems in their books and chatbots can be considered a part of a CCM solution perfectly fine. There are plenty of other tasks and applications for NLP as illustrated in the below figure 6 by Gupta et al (2020, chapter 1).

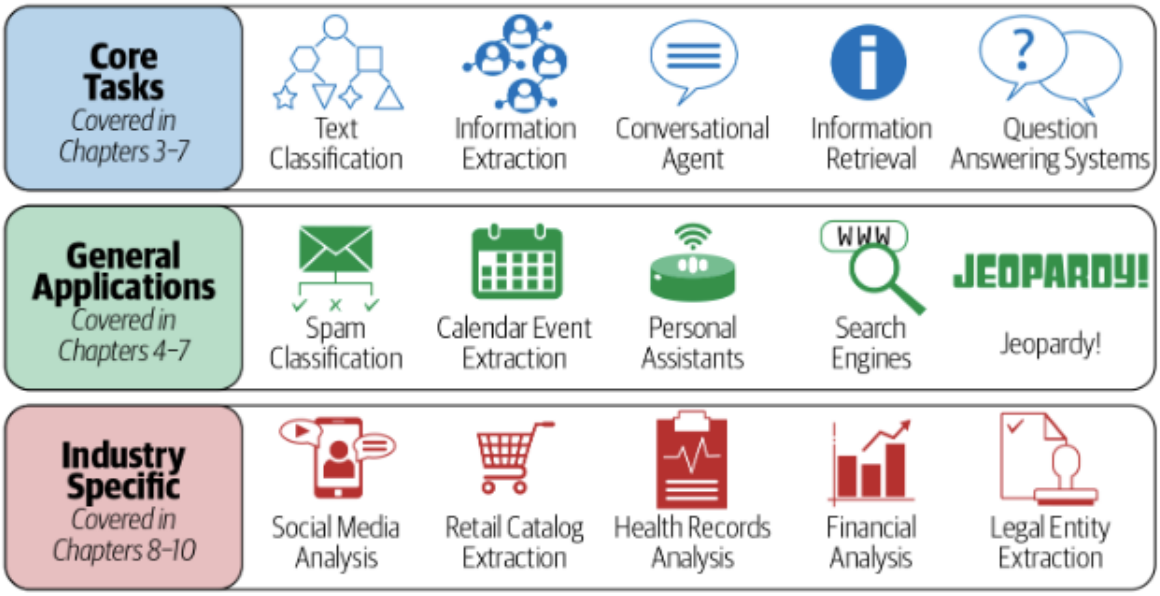


Figure 6. NLP tasks and applications (Gupta et al, 2020 chapter 1)

ML and NLP are quite closely related according to Gupta et al (2020, chapter 1). Supervised and unsupervised learning methods are also applied in NLP systems. Gupta et al (2020, chapter 1) provides spam classification as one of the sample use cases for NLP where instead of pure ML models are used, NLP specific learning models are used. NLP technologies are used to understand words and their contexts from the contents of emails and to classify the email as spam or non-spam. As ML supervised learning, NLP supervised learning techniques also use labelled data for the training data while unsupervised NLP learning works with unlabelled data, just as unsupervised ML learning in general. The algorithms used in NLP are slightly different to pure ML, although both work with transforming data into numbers and learning patters from the numeric representation but deeper technical details go beyond the purpose of this study. (Gupta et al 2020, chapter 1).

2.3.2 NLP application development

I will limit this chapter to NLP applications done with ML approaches. There is a growing potential on deep learning technologies and NLP (Gupta et al 2020, chapter 1) but as deep learning is not part of this study, those technologies are subject to possible future studies.

According to Gupta et al (2020, chapter 1) developing an NLP model for both supervised and unsupervised ML approaches for follow three common phases:

- Feature extraction from text.
- Learning a model from feature representation.
- Evaluation and improving of the model.

Feature extraction, as mentioned in the previous chapter, is a task where text and words are transformed into their numeric representations. That phase is followed by selecting an algorithm to learn from the numeric representation of the original words. Finally, the model is evaluated typically from two different perspectives, model accuracy and performance and business impact. (Gupta et al 2020, chapters 1 & 2).

A slightly more detailed process diagram for NLP system development is described in the below figure 7.

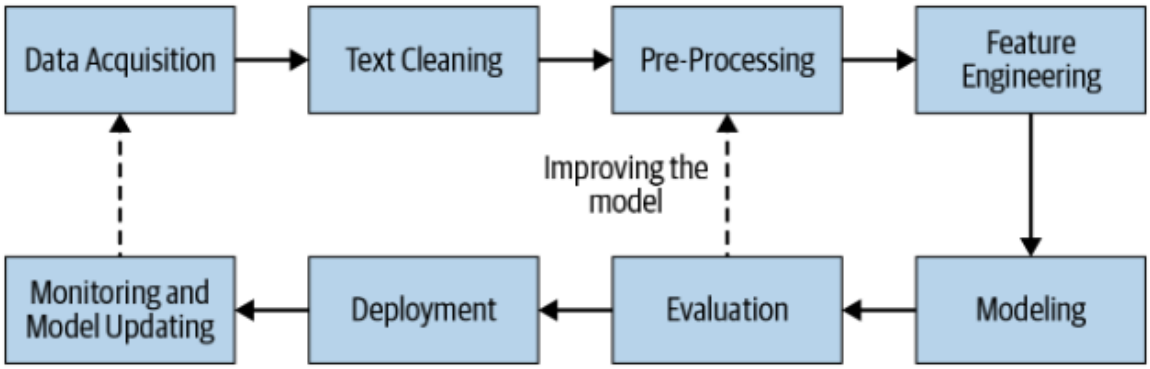


Figure 7. Generic NLP pipeline (Gupta et al 2020, chapter 2)

Data is the main driver for any ML and NLP system and therefore also NLP system development begins with acquiring the data. In an ideal situation there is already labelled dataset available for the development, but there are also techniques to generate that data if needed. Text cleaning is a step of the development process, where non-textual data, such as metadata and markup, is removed from the dataset. Spell correction can also be applied at this phase of the process. Outcome of Text Cleaning phase is a standardized text which can be more optimally utilized by the following steps. (Gupta et al 2020, chapter 2).

Pre-Processing is a phase where the cleaned text is further processed into a format which NLP algorithms can utilize more effectively. Typically, NLP software work with sentences and for that preprocessing is used to split the cleaned text into words and then to sentences. Depending on the need, further standardization of the text can be done at this phase. This standardization typically includes actions like turning all text to lowercase and removing non-essentials words, such as stop

words (such as “a”, “an”, “the”), punctuation and/or numbers, from the text. (Gupta et al 2020, chapter 2). However, Hapke et al (2019, chapter 2) do state that stop words can also include useful information, and the specific use-case determines what pre-processing steps should be applied. This is learned through experimentation which also Gupta et al (2020, chapter 2) highlights. Stemming and lemmatization processes can also be applied. There are other techniques available, all aiming to create the best possible dataset for the NLP application. (Gupta et al 2020, chapter 2).

Feature engineering, or sometimes referred to as feature extraction, is a step where the cleaned and pre-processed text is transformed into numeric format which the ML algorithms can effectively utilize. Once transformed, the modelling phase of the pipeline can be started. Depending on the need there are multiple ways to build the model. Simple rule-based heuristic approaches can do quite well with simpler needs while more complex needs typically require more complex modelling solutions such as stacked models, where multiple different models are being used. In the stacked model, multiple models are used to build a metamodel which is then used to create the final prediction. This is visualized in figure 8 below. (Gupta et al 2020, chapter 2).

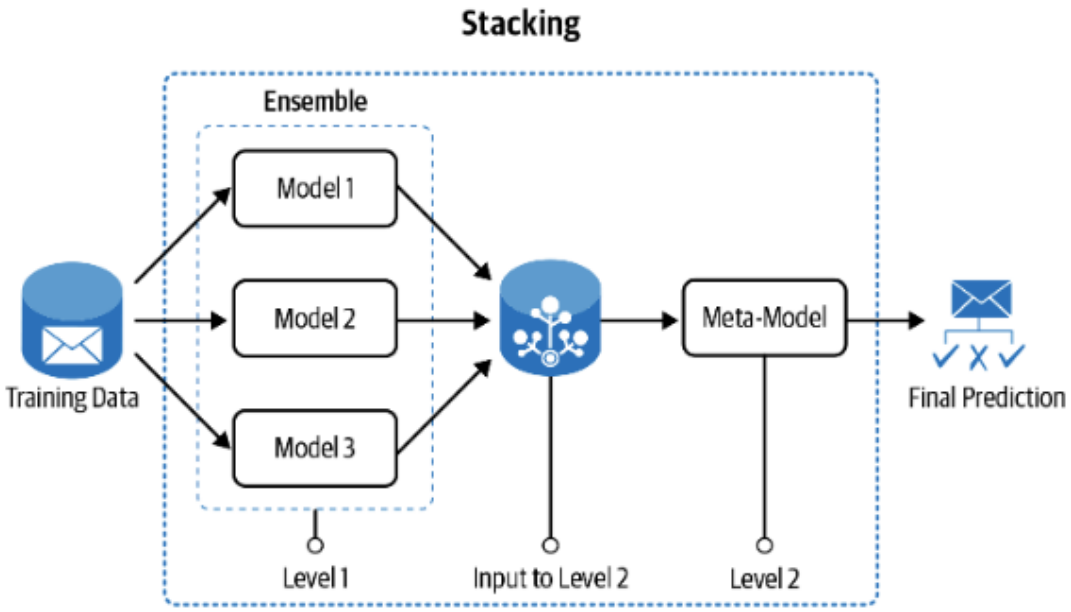


Figure 8. Model ensemble and stacking (Gupta et al 2020, chapter 2)

Modelling phase is then followed by the “Evaluation” step which is one of the key steps of the pipeline. In this step, the system’s performance with unseen data is being measured. Gupta et al (2020, chapter 2) separate evaluation into intrinsic and extrinsic evaluation types, where intrinsic evaluation focuses on intermediary objectives and extrinsic into final performance of the system. Hapke et al (2019, chapter 1) also name evaluation of the model as one of the key steps, but they do not

differentiate intrinsic and extrinsic types, although they do mention business goals and precision scores as one of the metrics.

For intrinsic evaluation there are multiple different metrics for the performance. For typical NLP systems, precision and recall are the most often used metrics. Precision-metric is used in various classification use cases where mistakes in positive class are more costly than mistakes in negative class. The metric defines how precise and exact the model's predictions are and it specifically measures how many of the case of interest it can classify correctly. Recall-metric is a complementary metric for precision-metric, and it also used for classification use cases where positive results are more important. Recall measures how many of the classified predictions are indeed correct. (Gupta et al 2020, chapter 2).

Extrinsic evaluation is more about evaluating how the NLP system meets the business expectations of the initial need. In the end, the business representatives don't typically care or necessarily understand much about the intermediary metrics if the NLP system is indeed solving the business need well. NLP algorithms don't offer metrics for extrinsic evaluation as such and that is typically more expensive process and requires the organization to define the metrics and follow them up. (Gupta et al 2020, chapter 2).

Evaluation is then followed by deployment and finally monitoring and model updating steps. In deployment, the NLP system is taken into use for the actual business need. Typical NLP module is deployed as a web service, where a larger system uses the NLP module to do whatever it was designed to do and then uses the output from NLP module to perform different tasks or to continue its processes. Monitoring here refers to evaluating that NLP module is staying consistent with its results and making sure it can handle bigger loads of requests if that kind of scenario should happen. Model updating is done based on the results of monitoring and in a scenario where new datasets have been gathered and would be available for retraining the module. (Gupta et al 2020, chapter 2).

2.4 Customer Communication Management

This subchapter describes more in detail what CCM solutions are like and how a typical document creation flow looks like. In general, CCM is about processes, methods, and tools around customer communication (Nelms & Phifer 2023). It is important to understand that CCM is much more than just creating and delivering documents. In this study however, as described earlier, I am focusing more on document creation and its distributions as it is ENIT's core business.

In the Figure 9 below there is a simplified process flow diagram on how a single transaction (i.e., document) is processed within the CCM and its surrounding solutions.

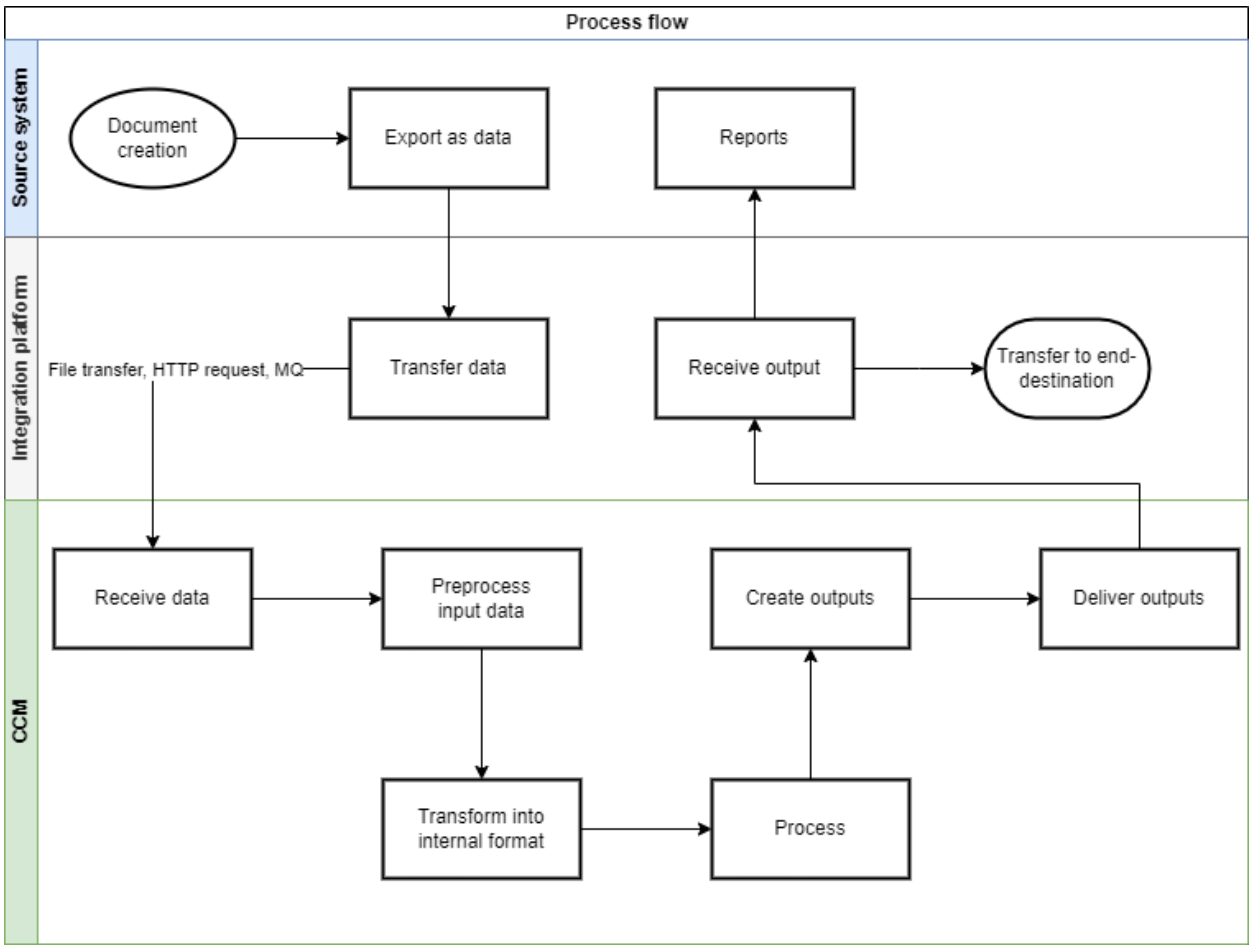


Figure 9. A single transaction process flow

The process for a document creation starts in the customer source system which can be any system. In our typical setup though, the source system is a CRM, ERP, or a billing system. Let us use an invoice as a document example when describing this process for simplicity reason, but in general the same process flow applies to most customer documents.

The typical transaction processing flow consist of the following steps:

- 1. CCM solution receives data from another system.
- 2. Input data is parsed and mapped into an intermediate format which the CCM template creator will use.
- 3. Different elements in the template (e.g., customer name and address) are populated based on the input data along with fixed elements such as company logo and name.
- 4. Final output, for example a PDF, is created and delivered to an endpoint (e.g., email server, self-service channel, internet bank, archive etc.).

In the generic scenario, all the above process phases happen automatically in runtime and no manual actions are needed. Depending on business rules, some documents can of course be halted, and not to be sent to end-customer before separate approval, but in general the process is fully automated. There can also be modifiable documents, where the process is slightly different but for simplicity point of view that is not necessary to be described here.

Input can be delivered to the CCM system in many ways. Transferring the data over internet as Hypertext Transfer Protocol (HTTP) requests is becoming the new normal in CCM but moving files via File Transfer Protocol (FTP) and Secure File Transfer Protocol (SFTP) or similar file transfer protocols are still widely used. Input data can be in any format based on the sending system. Typical file formats are fixed position-based text files, comma separated value (CSV) files, extensive markup language (XML) files, JavaScript object notation (JSON) or basically anything. The input data is parsed to an intermediate format (CCM internal) which is optimized for the template creator for simplified logic and improved performance. Output endpoints are based on the business requirements as is the format of the output. Typical output formats and destinations are text for Short Message Service (SMS) messages, HyperText Markup Language (HTML) for emails and internet portals, Portable Document Format (PDF) files for archives and email attachments and text, XML, CSV of JSON for data warehouses or any other reporting tools or databases. There are of course endless possibilities which are not necessary to be explored in this study, but this should give a brief overview of what a CCM solution is typically creating as outputs.

From development point of view, major implementation effort takes place behind-the-scenes for phases two and three which we can call the input data mapping phase and the template design phase. In the input data mapping phase, the input data is analysed and mapped into a format which the template design tool can effectively use. In template design phase the output (e.g., invoice) template is designed and input data field and references are mapped to their corresponding places. In addition to the variable data, some fixed elements such as company logos and contact information is typically added to the document. Then input data mapping and template design modules are taken into use by deploying them to an application server, then the software is tested and eventually deployed to a production system where the actual use of the template can start for the end customers. The process is similar than in any software development process when it comes to development, testing and production usage.

In addition to the data conversion and template and output design (Dev), CCM solution production usage includes monitoring and operative actions (Ops, or AMS, application management services). OPS work includes monitoring the production environment for possible issues and other tasks as needed or agreed. Many of ENIT customers handle this approach with a combined team of

development and operational specialist forming a DevOps team. However, from the topic of point of this study, operative details and recommendations are out of scope for this study.

3 AI use cases in CCM

In addition to the theoretical framework knowledge gathering, to better understand what AI features ENIT should to their product offering, I decided to host a brainstorming session among ENIT employees to discuss use cases where AI could improve CCM solutions and where it would be most helpful. The brainstorming session here acts as the main source data collection.

In my opinion brainstorming is a great way to gather ideas around a topic and solve problems among competent and experienced professionals. Since Moilanen, Ojalahti and Ritalahti (2022, chapter 4) describe brainstorming to be a good method for collaborative problem solving I thought it would be extremely beneficial to utilize my colleagues' expertise to gather up their ideas while also possibly reinforcing my own ideas of the possible AI appliances. I prefer continuous discussion over conducting a survey, although a survey could have been a great source of data as well.

I acted as a facilitator of the brainstorming session during ENIT's spring kick-off meeting in Norway in March 2023. The facilitator should create a relaxed and open atmosphere while preserving a goal-orientated focus without affecting the discussion too much (Moilanen et al, 2022, chapter 4) and I believe I achieved that well. As facilitator, it was most natural to me to keep in mind the "researcher's lens" method and not affect the discussion too much. Our brainstorming group included seven volunteer members of our team in addition to me. The group members should have proper skillset, attitude, and abilities to work on the task at hand (Moilanen et al 2022, chapter 4) which I believe the group members fulfilled. However, I do not know their competence or knowledge level on AI capabilities, but as the session was a volunteer event for colleagues who were interest in the topic, I like to think they joined with some background knowledge on AI. Moreover, while their knowledge in AI might have been limited, their extensive knowledge within the CCM domain should still be beneficial and should help them understand where AI can help. I thereby do not see big concerns in this regard.

The original handwritten notes of the session can be found in Appendix 1 "Brainstorming notes". Those notes are mostly bullet points and further content analysis and elaboration is found from this chapter.

I will describe each discussed topic in the brainstorming session first in detail and then analyse it within the context of AI, in specific for compatibility with ML and NLP models. I will also try to analyse the potential value to ENIT and possibly for the whole CCM industry. For some of the points, analysis also includes the possible complexity of the implementation of the identified feature, and its feasibility as the use case for the prototype creation. The more interesting and feasible topics I will describe more in detail while others are only briefly described. Some of the topics are not

feasible as AI initiatives for ENIT for now as I will explain for the applicable topics. Certain terms in the topics are widely used in CCM business terminology but may require elaboration for someone coming from outside of CCM and are therefore described here more in detail.

The objective of this chapter is to identify the “low-hanging-fruit” use case, i.e., the task with most benefits without too much risk and without too much effort (Eden & Long 2014, Introduction). That use case will be foundation for the prototype implementation.

3.1 Brainstorming topics

The following ideas were brought up in the brainstorming session and further discussed:

- Log analysis
- Scaling
- AI supported migration.
- Email tracking and statistics
- Help with template design process.
- Documentation
- Test case identification
- Troubleshooting and automated fixing
- Adhoc documents

3.1.1 Log analysis

All ENIT supported CCM solutions create vast amounts of log files and data. Simplified put, log data is anything the application outputs in text format, including but not limited to processing timestamps, document details, and anything the software developer or even customer might have specifically wanted to log for whatever reason.

During the discussion and afterwards, log analysis as a topic offers two different approaches for AI implementation. First approach is to classify log entries and the other is to learn from it. Classifying log entries seems like a “text-book-sample” of supervised machine learning model while learning from the data is more unsupervised and possibly deep learning use case. As deep learning is not part of this study, I will focus more on supervised learning and unsupervised learning possibilities for log analysis.

ENIT has a product which monitors applications for errors and other possible issues. Currently it does not have any AI built into it. The product monitors the application l in real time and notifies relevant people or entities by email or other agreed method of communication whenever it detects a transaction with error or warning status. Before joining ENIT, in one of the projects I created log

monitoring application where the notification creation was based on keyword detection. Whenever some of the keywords, for example “error”, “failure” or “unable” was found from the logs, a notification in a dedicated Teams channel was created for the operations team with the detected keyword and its log entry details.

One of the issues we face with our log monitoring tool, and which also I faced with my own implementation is that both solutions create too many unnecessary notifications and alerts. The transactions with “warning” or “error” statuses are not always something that require action. My keyword-based-log-monitoring-tool also triggered notifications for the operations team for log entries like “No error”, which is completely unnecessary if detecting errors is the target.

As described in chapter 2.2. ML could be used to classify data. Following on that idea, supervised machine learning could be utilised here to classify the entries picked up by our log monitoring solution. By applying proper classification model to the entries, unnecessary notifications and alerts would be minimized and therefore helping to identify the notifications that are of greater importance. There are plenty of data available to train a ML model to do a preliminary classification of the entry, before sending it to the recipients. This seems like a solid idea for ENIT to develop and could be considered as the scope of the prototype.

On the other hand, unsupervised learning could be used to learn patterns and other information from the data. I will discuss the uneven data loads of typical CCM solutions a bit more in the upcoming *Scaling* subchapter but that is indeed information that could be learned also from the log file analysis. Anomality detection should also be achievable with unsupervised learning which should help with overall quality of the solution. Unlike the supervised learning classification example described earlier, I cannot come up with similar “quick wins” from the unsupervised learning data insights. ENIT should first identify what they are looking for and whether that data is indeed providing added value. I would prioritise the classification of possible errors over the unsupervised learning log file analysis exercise for now.

3.1.2 Scaling

Scaling here refers to capacity of hardware resources, more specifically computing power and memory in servers and sometimes even hard disk space. With ENIT customers, the data flow is hardly ever constantly the same throughout the month. Instead, there are typically some “peak days” with much more data than in the rest of the month. This is especially common with invoicing documents, where typically companies tend to send most of their invoices on specific days. These “invoicing” days require much more computing power from the solution and platform compared to the rest of the month.

Scaling can be used to optimize the hardware resources capacity, namely have the correct amount of capacity for the processing need. Scaling is especially useful when using cloud services where it is possible to easily adjust the capacity needed. (Barr 2018). When using traditional on-premises server to host the solution, it typically is not feasible to change any of the hardware specifications regularly. In on-premises server setup, the capacity is typically set up to match the most demanding scenarios, for example the business-critical invoicing days, to ensure feasible processing times for those documents. Typically, this means that the server is running with at least some idle computing capacity for most of the month. For on-premises setup, I do not believe scaling would benefit ENIT or the customer company.

However, AI could help to identify the patterns on bigger computing capacity demands on the solution. Unsupervised learning should be able to identify patterns on higher volume peaks (invoicing dates for example) when given enough data. That data could be used to predict when the demand for capacity is higher and therefore prepare for it by scaling the environment if it is running in cloud. Cloud service providers do offer their own services for autoscaling as well though (Barr 2018) but with added cost.

While AI driven scaling appears to be quite beneficial, it really thrives on solutions and platforms build on cloud solutions. On-premises servers capacity adjusting isn't really feasible in CCM business, as adjusting memory chips, CPU units or disks always also mean downtime for the solution and typically CCM solutions should be up and running constantly. On virtual machines capacity adjusting is more feasible to do, as the resources are easier to adjust, but it requires planning and typically also manual actions to get those adjustments done. Cost-savings and other benefits for more fit-for-purpose computing capacity are however quite limited, especially as the initial resource capacity is usually calculated for the highest demands and the upfront investment for that is already done.

Effort to create an unsupervised learning module to learn capacity demands is not probably too considerable, but the benefit and usability of that model is mostly limited to cloud solutions. However, since cloud service providers already offer automated scaling solutions, ENIT should bring some additional value to their offering which will increase the efforts needed to create that model. Therefore, I would not consider scaling as the top priority to focus on now for ENIT.

3.1.3 AI supported migration.

As described in the earlier chapter, one of the most typical CCM business cases is creating a document template, such as an invoice or and order confirmation, for any specific business need. *Migration* here refers to customer company changing from an existing CCM system to a new one

where ENIT is providing the document template creation service. As digital transformation continues, we have seen that many companies look for alternative and more modern solutions to handle their document creation and distribution. This is one of the core business cases for ENIT but there is no AI helping in this currently.

Creating the document template is a complex and time-consuming process. Templates typically contain both variable and constant data where the variable data is use-case-specific while constant data are company or template specific texts. This constant data could include for example labels specifying what the variable data stands for. In the below table 1, I am simplifying the difference of these to reduce the change of misunderstanding.

Table 1. Constant data versus variable data sample

Contant data/label	Variable data
Customer number	12345678
Document date	1st January 2023

When creating the template one of the tasks is identifying these data types and this is something I think AI could be used for. With migrations, there are already existing templates available from the previously used solution and AI could be used to analyse those templates. Unsupervised learning should be able to differentiate repeating data from the customer specific data and therefore help with the process of identifying labels from variable data. This classification should be useful when creating the actual template.

In a typical template development scenario, the development also included identifying where in the input data the specific variable data is found and where it is shown in the actual template. This is should also be something AI could be used for. There are ways to extract data from images and PDF files (Gupta et al 2020, chapter 2), so an AI module or similar available software libraries could be used to find the same data from the input data. With enough sample data, identifying variable data from the extracted data set should be quite feasible while at the same time analysis of the constant data should be achieved.

Some of the CCM solutions to which ENIT has been specialized in, stores the layout definition files in a predefined JSON format. These definitions include most, if not all, of the information requires to create the final output for the layout, including but not limited to positions of each text and image element and rules on the content of those text or image elements. Since that data can be made available with the data extraction step described earlier, it should also be quite feasible to create at least a preliminary layout definition file by mapping the extracted data to the predefined JSON

format. For the developer, this would mean a baseline template design definition file to start the work on, instead of starting with an empty one.

Having AI enabled automation with the migration process would drastically reduce the time-to-market period. However, I believe this requires quite a lot of effort and while it can be considered high value it cannot be considered as a low-hanging-fruit due to its complexity. I would not consider this as the first step into ENIT's AI journey but something that should be evaluated further. From marketing point of view, AI enabled template design sounds exciting and as far as we know, none of the other providers offers this yet.

3.1.4 Email tracking and statistics

One of the key questions the business representatives typically have is whether the email sent by the CCM solution was indeed delivered successfully to the end customer, and whether the customer has opened the email. This is especially valid with invoices, as the invoice is often the way for the company to get the payment for its services, but it is most likely valid for offers and tender letters just as much.

Traditional email services do not really track the emails' delivery more than whether the receiving email server received the email. For unknown or invalid email addresses, and in certain error scenarios, the receiving email server responses with a "bounce" message with an error code, which the sending email server typically delivers back to the sender indicating that the email in question could not be delivered (Finch 2005). So, the traceability of the email delivery when using normal email servers is generally quite poor.

There are certain email services which offer better traceability. One to mention is MessageBird Email (formerly SparkPost) which offers which ENIT also uses in its ENIT Advanced Email service. Technical details on how the tracking performed is beyond the scope of this study. MessageBird email offers reports and statistics through their service.

Hence, I do not consider "Email tracking and statistics" as a matter of AI interest for ENIT as there are existing solutions for it and I cannot think of any simple additional benefit that AI, ML or NLP could bring to it. Maybe unsupervised learning could learn some patterns on it, but I do not see that bring significant added value to the existing solution.

3.1.5 Help with template design process.

Template design process was described in the earlier section as part of the *AI supported migration* point. The same points described in that section apply here. AI could be utilised in analysing the input data and sample template files and therefore helping with the design process.

Otherwise, NLP modules could be utilised for spell checking and evaluation of the quality of information presented on the template. But often, ENIT is not responsible for the texts or otherwise contents of the templates. Instead, these texts are defined by the customer or by regulations, so ENIT has limited possibilities to affect these texts. Regardless of that, ENIT could investigate adding NLP modules into its products which are meant to generate and maintain texts. However, I see little added value for this for now, except for the common points described in the *AI supported migration section,* and therefore would not consider this as the first AI implementation for ENIT.

3.1.6 Documentation

AI code assistants can be used to help documenting code (Morris 2023). Having worked in software development line of business for more than ten years now, documentation tends to be the part of the process which is done without too much effort if at all. Maybe it is not that exciting as working with the code, but nevertheless it is not step that should be neglected. Proper documentation helps understanding the solution and improves collaboration possibilities (Morris 2023). ENIT has decent level of documentation for their solutions and products, but I would be exaggerating if I said it would be perfect.

While documentation is indeed important, I would not focus on this part of AI usage for ENIT just now. ENIT is looking into AI code assistants at the time of writing this, so maybe the AI powered documentation need will self-resolve itself in the becoming months with the AI code assistant tools. For this study, I will not investigate AI powered documentation more.

3.1.7 Test case identification

Testing is an important phase of any software development and test case identification is part of that process. According to Gupta et al (2020, chapter 2) extensive testing is a mandatory to be executed before taking any software to actual production use. Identifying the simple and obvious test cases is usually quite low effort task, but identifying all needed test cases is much more demanding task.

Morris (2023) lists many possible AI code assistants, which can help with creating test cases. ChatGPT also provides this feature, but providing complete code to OpenAI’s product is something

they strongly recommend against as they will use that any data provided to ChatGPT to improve their offering.

Following the point I made in the previous topic, AI powered test case identification can self-resolve itself with ENIT's ongoing initiative for AI code assistants and therefore I will not consider this as a point of interest for the prototype nor investigate it further in this study.

3.1.8 Troubleshooting and automated fixing

The idea here would be having an AI module monitoring the logs for possible errors (much like *Log analysis* section described earlier), identify what causes the error, apply a fix in the code or wherever applicable, deploy the fix and rerun the failed transaction or task. All of this would happen automated without human interaction.

To some extent, I can see this possible but only if the error is caused by something trivial, for example a typo in code or a script, which the AI module would identify, fix, and then deploy. Anything more complex is beyond what I can imagine AI is currently capable of. Even for the simpler case, the AI should be monitoring the log, identify the piece of code or script which is causing the error, fix it while making sure anything else doesn't get broken, preferably also test it, and then deploy it. Depending on the solution, rerunning the transaction can also be complicated.

While this sounds amazing and would help tremendously, I am afraid that the complexity of the needed AI module is beyond my comprehension at this time making this topic therefore too complex to be considered as the use case for the prototype. However, this is something that ENIT can investigate later after initial simpler AI implementations are done, but not as the first thing.

3.1.9 Adhoc-documents

Adhoc-documents here refer to documents which will not go through the standard processing flow of a CCM solution. The term "adhoc" here comes from earlier product by OpenText which was used to handle these kinds of cases. In this scenario there are two different use cases which this topic is about. In the first one the document is halted by some predefined business rule and requires a human interaction before proceeding with the delivery. The other use case is a modifiable document where again, human interaction is needed to finalize the document before it can be delivered. The difference between the two use cases is that the first one is halted during the runtime processing where the other is not reaching the runtime processing before it is finalized.

A sample use case for the first case is an invoice halted during processing as its total amount is above the set business rule. A human interaction is required to verify that the invoice is indeed

correct, and it can be delivered to the customer. In many of our customer solutions, we have developed a free-text section on the layout where the representative can enter text if needed. This is where the AI could be used for. AI could analyse the content of the invoice and offer text options created with an NLP module. The same analysis could also be used to check whether everything on the invoice seems to be correct. The human representative could then save time and use the analysis from the AI module to decide on the possible actions for the invoice. NLP modules could also be implemented to enable spell checking and understandability of the text. These tasks seem rather straightforward to implement and could be of interest to ENIT and to the whole CCM industry.

The second use case, where the document finalization happens before the runtime processing, is typically a scenario where there is a premade base template with customized content. While this can typically be achieved in a runtime solution as well, these templates offer the flexibility with using premade text sections while allowing modifications to them if needed. Some of the examples of this kinds of templates could be insurance claim responses where the base template is the same, but the final document depends on the claim contents and on the approval or rejection of the claim request. These premade text sections can include the basic approval or rejection texts which the representative can then modify as needed. Much like with the first use case, NLP modules should help here with spell checking and understandability. If the finalized content is saved, that data could be used to teach an NLP module to offer possible responses and further help the representative to save some time with similar activities in the future.

While both use cases seem very prominent for AI appliances, they are only a subset of all documents going through a CCM solution and not all companies are using these features. Also, I believe the complexity of the implementation is beyond the scope of the prototype work here. However, I believe that this should be one of the first AI applications ENIT should focus on as it can bring significant benefits for the end users with improved quality on the text and reduced effort on finalizing them. With GPT models and Generative AI modules becoming more available, I believe implementation of these kinds of features will require less effort. However, as this study focuses on ML and ML, this will be something to be considered later.

3.2 Summary and prototype selection

I realize that there can be other use cases for CCM AI initiatives but to limit the scope of this study I decided to choose the prototype implementation based on one of topics brought up in this chapter. I believe the vast amount of CCM expertise my colleagues have is valuable and their topics on where AI could help should not neglected.

In the below table 2 there is a summary of use cases described in this chapter and their “effort to create”, and estimated value for ENIT. Ratings are based on my studies and observations on each topic, described more in detail for each item in earlier in this chapter. Ratings are low, medium, and high.

Table 2. Use case analysis scoring

Topic	**Effort**	**Value**	**Notes**
Log analysis	Low	High	
Scaling	Medium	Low	Value higher in cloud solutions, but cloud vendors already provide their own.
AI supported migration	High	High	
Email tracking and statistics	Medium	Low	Existing solution without AI solving most needs
Help with template design	Medium	High	Similarities with AI supported migration
Documentation	Medium	Low	AI code assistant initiative might solve this
Test case identification	Medium	Medium	AI code assistant initiative might solve this
Troubleshooting and fixing	High	Medium	
Adhoc-documents	Medium	High	

Following the low-hanging-fruit definition by Eden & Long (2014), Log analysis -topic seems to be low in effort and high in value. Especially the supervised learning approach for classifying log entries should offer immediate value to ENIT.

Therefore, the prototype implementation for this project will be a ML classifier for log entries.

4 Prototype creation

This chapter describes the building of the log entry classifier prototype identified as the low hanging fruit in the previous chapter. As described earlier, ENIT already has a tool to monitor the application log files and gain information from possible error cases through that. However, this tool does not have any AI features in it. The prototype should serve well as the initial step having AI as part of ENIT's product offering.

The high-level idea of the final version of the ML classifier is to analyse each event picked up by the current monitoring tool and have the ML classifier decide whether the specific event requires human interaction. As mentioned before, one of the issues with the current solution is that it creates too many notifications, many of which do not require human interaction This prototype aims to create a concept to mitigate the number of unnecessary notifications by identifying the cases where the notification is not needed and therefore reducing the number of notifications overall and ultimately improving efficiency and user experience.

The current simplified process flow is described in figure 10 below. Any transaction with warning or failure status will generate a notification and gets stored in a repository.

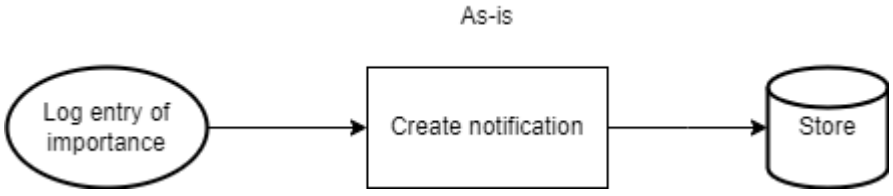


Figure 10. As-is process flow

By adding the ML classifier, the process flow changes a bit as described in the figure 11 below.

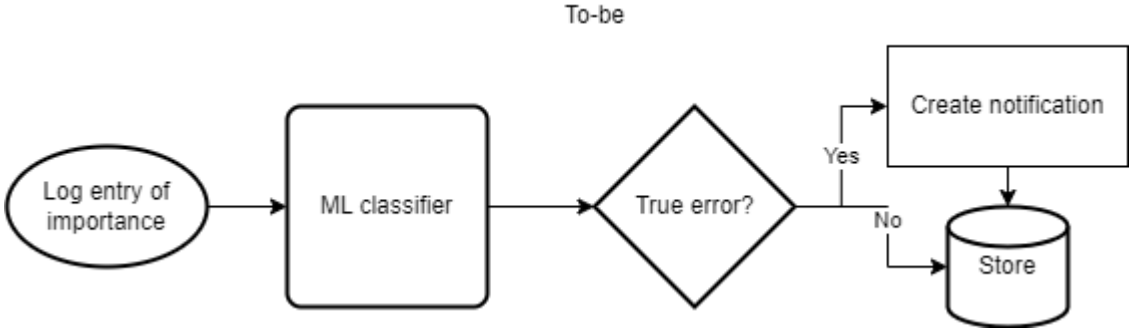


Figure 11. To-be process flow

Instead of creating notification for each log entry identified, the entry would be analysed by the ML classifier and only then the notification would be created if classified as true error. However, all the entries would be stored in database for easy access to enable possible retraining of the classifier model and continuous learning. This version of the prototype does not include elements of reinforced learning, where the user could report that the notification classification was not correct but that would indeed be a prominent future feature to be developed. That kind of advanced functionality is beyond the necessary scope of the prototype, although it is an intriguing idea. Having that would require changes in the current user interface of the solution, and creation of receiving the endpoint for retraining of the model, which is not overly complicated but will require considerable effort and is therefore left out of this study.

Also, as this is a prototype, this will not be implemented to the current product itself as such, but rather offers the concept of ML classifier's benefits. As each customer CCM solution outputs slightly different errors and data, I believe the most accurate ML model would require to be built with customer specific models with customer specific data. However, the concepts of this prototype should illustrate how that module would function.

Moreover, I tried to keep this chapter in not-too-deep-technical level but some of the parts do go beyond the normal business understanding requirements with the provided code samples. However, I felt it is important to describe the process in enough detail and I feel quite confident that I found the correct balance.

4.1 ML model creation

Combining the gathered knowledge in chapter two from Sandramouli and Das and Gupta et al, creating this solution should be a three phased process as described below.

1. Data acquisition and preparation.
2. Model creation and evaluation
3. Deployment and monitoring

4.1.1 Data acquisition and preparation

For the prototype I used application log data from one of our development servers. A sample log entry from that data is found below:

```
0809 082212 (1542) 2 Loaded items(number, from):2485, .\custom\ta-
bles\language.sls
```

The two first digits represent a month followed by date, represented also with two digits. Therefore, *0809* here means 9th of August. The next six digits is hours, minutes and seconds all represented

with two digits. The following value in the brackets is a log message identification number. The “2” after the brackets is log severity level. The value two (2) refers to warning level entry. The log entry severity level is followed by the actual log entry.

I started with around 100 000 rows of log data. Most of it is information and other data which is not relevant for the log monitoring tool. Instead of data that ENIT’s current monitoring tool uses, I used keyword-based search and separated all entries with “error”, “fail” and “unable” to a separate dataset from which I also removed any identical entries, as identical entries are more useful for ML learning algorithms based on the theoretical framework knowledge. That dataset ended up being around 1000 rows.

For supervised learning, the ML module needs labelled data (Sandramouli & Das 2018, chapter 1). I analysed the data in my cleaned dataset and labelled all entries with either “serious” or “ok” labels. Even though the below log entry included “Failed” text which typically indicates an error, in this application that is “error” which does not require any human interaction. I labelled the following log entry as “ok” hoping that the ML classifier will identify similar entries as not serious errors and therefore only storing them will be enough and no notification should be created.

```
0809 082241 (1113) 1 job end ERROR jobstatus 0 intjobid 55962 invoice number 0118417863
```

The below log entry is a “true” error, so I labelled that as “serious”. The ML classifier therefore should classify similar entries as “true” errors and create the notification in addition to storing them.

```
0809 135308 (1076) 1 XMLIN: XML parse error (message, line, column, position): "n", 1, 0, -1
```

After the labelling phase, I cleaned up the data as instructed by Gupta et al (2020, chapter 2) even though this is not an NLP module. I removed all other data but the log entry itself and from that I also removed punctuation and special characters. I will lowercase the entry in a later phase as that is easier to achieve with a scripting language in my opinion. After cleaning, the above log entry in the following format in the training set.

```
XMLIN XML parse error message line column position n 1 0 1
```

Probably the end of the entry, string “n 1 0 1”, could also be removed but for prototype creation I felt this was sufficient level of cleaning. After cleaning and labelling the data, I made it to a comma-separated value (CSV) file which will be used as the training set for the actual classification model.

4.1.2 Model creation and evaluation

The cleaned and labelled data in CSV format is used to create the training data set for the ML model. The file contains all the rows to be used to train the model with “ok” or “serious” as label separated by a comma.

Below are some above sample rows as they are in the file:

```
“XMLIN XML parse error message line column position n 1 0 1”,”serious”
“job end ERROR jobstatus 0 intjobid 55962 invoice number 0118417863”,”ok”
```

The data in the first position (separated by comma) is the log entry and the value in the second position is the label. All entries are separated by a line change making each entry as its own line in the file. In total there is 981 rows in the training set, which should be quite enough data for the prototype work.

The actual model is created with Jupyter Notebook. The complete notebook file is found in Appendix 2 log-classifier.md in ReStructuted Text format. Jupyter Notebook is a development environment for Python scripting which comes with great document-like user experience. I chose that out of no specific reason other than it is easy and flexible to use, and I have used it in the past for experimenting with Python code.

The first step is to read the CSV file with the log entries and their classifications (“serious” or “ok”) so they can be used for the ML training set. At this point, I will also lowercase all the log entries as I believe case sensitivity does not create any important difference with the entries. Data is loaded into Pandas data frame as that is in such format that most ML algorithms can efficiently use. Data is then split into Z and Y variables which is further split into training and test sets. Training set is the whole data entity while test set is a subset of the data. The size of the subset is typically 20 % of the complete data set. In the following steps, the ML model is trained with the complete data set and the test set (subset of the data) is used to test the model on its accuracy after that.

In the next step the training set is transformed into numbers (vectorized) as instructed by both Sandramouli and Das and Gupta et al. This is done by a module from machine learning library called scikit-learn. The data is then fitted into features and then created into an ML model. These steps are once again done with premade modules from scikit-learn library. The model is then tested against the test set created earlier evaluated with “accuracy” metric, which simply defined how many percent of times the classifier classified the entry in the test set correctly. I got 95 % percent accuracy with this algorithm which I am more than happy with. Below in figure 12 is the code snippet from Jupyter Notebook to illustrate the accuracy.

```
# using SVC (Support Vector Classifier) model
model = svm.SVC()
model.fit(features,y_train)

# test the model
features_test = cv.transform(z_test)
print('Accuracy score:', model.score(features_test,y_test))
```

```
Accuracy score: 0.9487179487179487
```

Figure 12. Model creation and accuracy testing

As the last step of the Jupyter Notebook actions, I need to export the model and the vectorizer into binary packages to be used by the classifier itself. As instructed by Gupta et al, ML modules should be deployed as a web service (2020, chapter 2). The training phase, which was implemented with the notebook, is not something that needs to happen in runtime implementation with each classification action. Instead, the training phase should be revisited when there is more and better data available, and especially if there is not previously available data available which the current classifier does not classify correctly. This falls more into the monitoring phase which will be described a bit more in detail in the next sub chapter. The binary export is done with *joblib* module with the following code snippet as described in figure 13.

```
import joblib
# dump models into binary to be called by the webservice
joblib.dump(model, 'trained_model.pkl')
joblib.dump(cv, 'vectorizer.pkl')
```

Figure 13. Binary extraction of the model and vectorizer

4.1.3 Deployment and monitoring

The ML classifier is now trained and ready to be taken into use. For the prototype testing, I created a web service which can be called from the log monitoring tool. The classifier web service monitors HTTP requests into specific port into which the log monitoring tool would then send log entries for classification. The log monitoring tool is expected to clean the data and only send the log entry itself (that is without timestamps, log message id and log severity level), which is then classified by the ML classifier to either “serious” or “ok”, with accuracy of 95 % as predicted during the training phase. Based on this classification, the log monitoring tool should then create a notification for “serious” classified entries. For the “ok” classified entries, the entry is only stored, and no notification is created. As stated before, as part of this study, no actual changes will be implemented to the

current log monitoring product itself. Instead, this is simply a prototype and works in conceptual level.

The web service is built with Python and using Flask library to host it as a web service. The full code I used to test the prototype tests with, is found in Appendix 3. Detailed description of the web service code is beyond necessity for this study, but some of its features in high level I feel is necessary to describe. The web service uses the binary packages from the training phase to do the actual classification. It receives a request with the log entry to be classified as the data. It vectorizes the character formatted log entry with the same vectorizer module, which was used to train the ML model, then submits the vectorized value of the entry for the classifier, and then returns the classified value of the entry to the requesting entity. In my test setup, I am running the web service as a stand-alone instance in my local setup. In a production setup I would recommend running it as a scalable microservice. However, the possible production setup details are beyond the scope of the prototype.

To test the classifier, I build a testing setup with one of ENIT's product called Flowbuilder. The Flowbuilder simulates the log monitoring tool in this setup. For testing purposes, I can set the log entry manually to whichever value I want and then send it to the ML classifier to be classified as "ok" or "serious". The ML classifier is running as a web service that the Flowbuilder setup will call to get the classification. In below figure 14 I have run some tests with the setup and on the right-hand side the values used for testing are listed along with the classification given by the classifier.

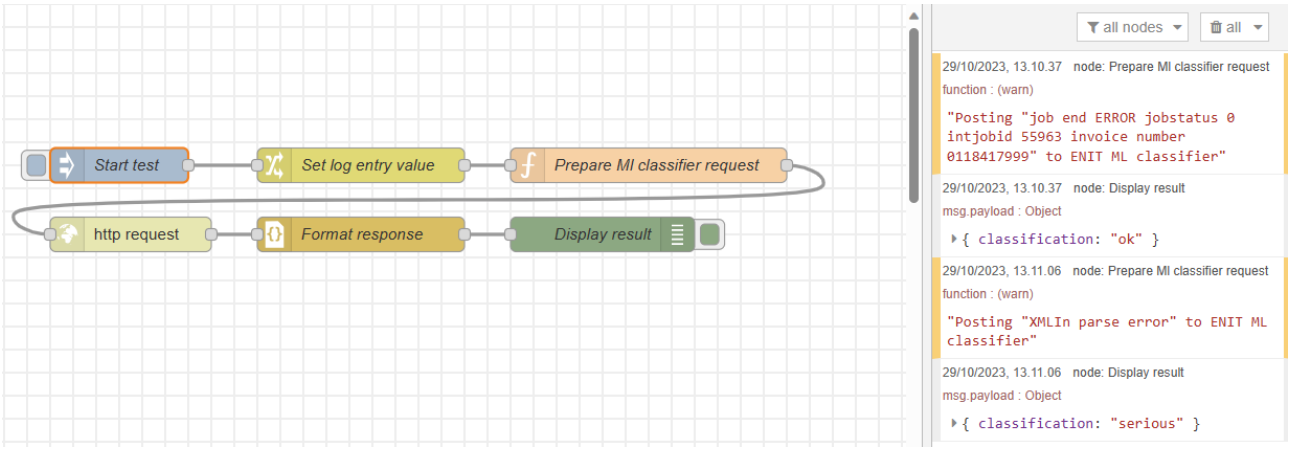


Figure 14. Prototype testing

As can be seen from the Figure x. The first simulated log entry error is `"job end ERROR jobstatus 0 intjobid 55963 invoice number 0118417999"`. It is important to notice, that this is not identical to the sample "ok" log entry described previously,

meaning that the classifier has never seen similar entry before. Nevertheless, the ML classifier classifies that as “ok”. This is precisely what I wanted the module to do.

The second simulated log entry is `"XMLIn parse error"` which, once again, is not identical to the “serious” labelled log entry described earlier. Yet, the ML classifier classifies it correctly as a “serious” error. I tested this with multiple other simulated log entries with high correct classification rate. However, some of the classifications were not labelled as “ok”, although I would have expected them to be. In the previous chapter, the accuracy of the model was calculated to 95 %, therefore this kind of behaviour is expected and falls into the error threshold. Also, I did not identify any tests that were true errors but would have been classified as “ok”. This behaviour is also as intended. From operations point of view, it is better to have false errors classified as “serious” than true errors being classified as “ok” and therefore those going unnoticed.

One of the things that Gupta et al recommended was to monitor is the performance of the classifier. Accuracy wise, the classifier is performing as expected and in perfectly feasible timely manner as well. Average response time from the classifier was around 120 milliseconds which is perfectly adequate for such a classification.

4.2 Prototype conclusion

The prototype described here is working as expected and I am confident that the concept of the prototype could be implemented as part of ENIT’s current log monitoring tool and serve well as one of the first steps in ENIT’s AI journey. For the production version and for the most accurate classification model, instead using the prototype’s training model, I would train the model with customer specific data while following the same approach as was used for the prototype model creation. This should increase accuracy as the training data is from that specific solution. I would also highlight the importance of storing the data and their classification values so that the model could be re-trained with more accurate data set.

In some time, with enough data from multiple customers, a common data model could be created which would cover the most typical log entries. Using one common data model for all the customers would be easier to maintain than customer specific models, and with enough data, it should be just as accurate or even more accurate.

The conceptual approach on this prototype is fully usable outside ENIT and can be applied in CCM domain, and, for similar log entry classification need in any business domain for that matter. The concept should function optimally in any domain, provided that the training data is specific to the business domain.

5 Closing words

This study started with the main objectives to find use cases for ENIT to implement AI in their CCM related product offering and to describe the methods and principles on how to implement such features. Although there are many AI branches available, this study was limited to ML and NLP branches as those seem the most prominent branches for CCM (Aspire 2023) and according to the research in chapter two, ML and NLP applications should be rather easy to understand and implement. Because of these characteristics ML and NLP should serve ENIT well as the initial steps into its AI journey. The main objectives were achieved with great success.

The main outcome of the study is that there are indeed many possibilities for AI appliances in CCM industry and therefore for ENIT as well. Many prominent use cases were identified and for the most suitable and feasible one, a prototype ML application was implemented. Following “member checking” principle for evaluating the results, the outcomes of the study were presented in ENIT’s fall kick-off seminar to the members of the brainstorming session and the main commissioner. Different phases of the implementation process were described in an understandable way with certain deeper technical details. Even though AI seems complicated and while certain technical details still elude my full comprehension, many of the identified use cases should not be overly complicated to develop. These use cases should be evaluated further by ENIT for possible development tasks.

The prototype creation was a success. It provides a solid indication that ML features can be beneficial for ENIT and specifically for one of its current products with rather low implementation effort and minimal risk. The study provides a common development process for developing ML and NLP applications which can be utilized in the practical development work helping ENIT to develop AI features into their other products.

One of the main challenges of the study was the limited availability of scientific research on topics combining AI and CCM. There are plenty of research and literature available for AI, ML and NLP domains but for CCM, the availability of such material is quite limited. Aspire and Gartner have studied AI possibilities with CCM to some extent, but I would not consider their research purely scientific, especially from the objectivity point of view, although I do not disagree on their findings either. Because of lack of CCM related research, the CCM aspect of this study is mostly based on my and ENIT’s observations and experience on how the CCM business domain functions in general. Although I believe our 400 years of combined working experience in CCM domain reflects the generic understanding on how CCM domain functions, perhaps someone with different background, especially regional, could perceive CCM domain differently and could have slightly different views. However, I believe that the AI use cases identified in this study are applicable for CCM

solutions globally and I certainly understand that AI use cases are not limited to the ones included in this study.

Another challenge for this study is the ongoing and, at times, exaggerated enthusiasm and “hype” surrounding the field of AI. By “hype” here I specifically mean the overall awareness and presence in media and social media around AI, the effort and investments which are put to AI development, and the pace in which AI is evolving. Some of the observations and ideas we had in our brainstorming session just six months ago have already been resolved with new AI features, such as code assistants and other branches of generative AI. It was challenging to study a subject that is in a rapidly evolving state and at times I felt that limiting the study to ML and NLP might limit the full potential of the study, especially as the possibilities in generative AI seem considerably intriguing. However, ML and NLP are easier to understand and implement, and while generative AI could be too overwhelming for ENIT to start its AI journey, I believe the limitation was justified and even helpful for ENIT and for the study. To the best of my knowledge, this study is the first scientific research, albeit in a form of a thesis, combining AI and CCM domains and it is my sincere hope that it will not be the last.

I certainly have learned a lot about the practicalities about ML and NLP application development and their possibilities. Although I had experimented with ML and NLP before, I know understand more why each of the step during the modelling is done. I would encourage ENIT to also investigate the possibilities in deep learning and generative AI branches which were not in the scope of this study. It could be beneficial also to investigate what AI services and products different cloud service providers already offer. Those would come with added cost but if they provide significant value, the return on investment can be better than developing own similar features.

Acknowledgements: thank you Peter Grill, CTO at ENIT, for acting as the commissioner from ENIT, for participating in the brainstorming session and for the ongoing support. Thank you Anders Dalmén, CEO at ENIT, for your continuous support for the work. Thank you for the other brainstorming session participants (in alphabetical order) Peter Adermark, Markus Anding, Dan Byström, Mika Friman, Joakim Gröndahl, Lisa Huge and Jyri Keränen.

Sources

Anyona, R. 2017. The History of Artificial Intelligence. Harvard University Blog. URL: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/. Accessed: 27 October 2023.

Aspire Leaderboard. 2023. Communications Experience Platform (CXP). URL: https://www.aspire-leaderboard.com/best-communications-experience-platform. Accessed: 10 September 2023.

Barr, J. 2018. App Scaling – AWS Application Auto Scaling – AWS. Blog. URL: https://aws.amazon.com/autoscaling/. Accessed: 17 September 2023.

Blochi, P. 2023. The 6 Branches of Artificial Intelligence. URL: https://www.linkedin.com/pulse/6-branches-artificial-intelligence-paul-blocchi/. Accessed: 13 September 2023.

Davenport, T. H., Brynjolfsson, E., McAfee, A. & Wilson, H. J. 2019. Artificial Intelligence: The Insights You Need From Harvard Business Review. Harvard Business Review Press. Boston. URL: https://search-ebscohost-com.ezproxy.haaga-helia.fi/login.aspx?direct=true&db=e076mww&AN=2003692&site=ehost-live&scope=site. Accessed: 16 September 2023.

Dartmouth. 2023. Artificial Intelligence Coined at Dartmouth. URL: https://home.dartmouth.edu/about/artificial-intelligence-ai-coined-dartmouth. Accessed: 16 September 2023.

DataFlair. 2023. AI and Machine Learning – Know the core aspects of the deadly duo. URL: https://data-flair.training/blogs/ai-and-machine-learning/. Accessed: 16 September 2023.

Eden, J. & Long, T. 2014. Low-Hanging Fruit: 77 Eye-Opening Ways to Improve Productivity and Profits. John Wiley & Sons, Incorporated. New Jersey. E-book. Accessed: 25 October 2023.

Elisbury, G. 1998. The Turing Bombe. URL: http://www.ellsbury.com/bombe1.htm. Accessed: 22 October 2023.

Endpoint Protector. 2023. EU vs US: What Are the Differences Between Their Data Privacy Laws? URL: https://www.endpointprotector.com/blog/eu-vs-us-what-are-the-differences-between-their-data-privacy-laws/. Accessed: 13 March 2023.

Esposito D. & Esposito D. 2020. Introducing Machine Learning. Microsoft Press. E-book. Accessed 12 March 2023.

Everdeen, K. 2017. What is Artificial Intelligence? URL: https://docs.google.com/document/d/1ExKyP800FSXX-YzAmioZnefwfEyBQ_gwGP8uQN19nr0/preview. Accessed: 10 September 2023.

Finch, T. 2005. Bounce address protection for email. University of Cambridge Computing Service. Cambridge. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=261ebb7536a73748b8a43b6dc4ee39cb25512bef. Accessed: 23 October 2023.

Gartner 2023. Definition of Customer Communications Management (CCM). URL: https://www.gartner.com/en/information-technology/glossary/customer-communications-management-ccm. Accessed: 14 August 2023.

Gartner. 2017. Gartner Magic Quadrant for Customer Communications Management Software. URL: https://www.gartner.com/en/documents/3584118. Accessed: 29 August 2023.

Gollapudi, S & Laxmikanth, V. 2016. Practical Machine Learning. Packt Publishing. E-book. Accessed: 12 March 2023.

Gupta, A., Majumber, B., Surana, H. & Vaijala, S. 2020. Practical Natural Language Processing. O'Reilly Media Inc. E-book. Accessed: 8 October 2023.

Hapke H., Howard C. & Lane, H. 2019. Natural Language Processing in Action. Manning Publications. New York. E-book. Accessed: 7 October 2023.

Imperial War Museums 2023. How Alan Turing Cracked the Enigma Code. URL: https://www.iwm.org.uk/history/how-alan-turing-cracked-the-enigma-code. Accessed: 15 September 2023.

Manning, C. 2020. Artificial Intelligence Definitions. Stanford University. Stanford. URL: https://hai.stanford.edu/sites/default/files/2020-09/AI-Definitions-HAI.pdf. Accessed: 16 September 2023.

McCarthy, J. 2007. What Is Artificial Intelligence? Stanford University. Stanford. URL: http://www-formal.stanford.edu/jmc/whatisai.pdf. Accessed: 10 September 2023.

Mitchell, T. 1997. Machine learning. McGraw-Hill Education. New York.

Moilanen, T., Ojasalo, K. & Ritalahti, J. 2022. Methods for development work: New kinds of competencies in business operations. Books on Demand GmbH. Helsinki. E-book. Accessed: 25 October 2023.

Morris, C. 2023. 14 Best AI Coding Assistant Tools in 2023. URL: https://www.elegantthemes.com/blog/wordpress/best-ai-coding-assistant. Accessed: 22 October 2023.

Nelms, T. & Phifer G. 2021. Gartner Market Guide for Customer Communication Management. URL: https://www.gartner.com/doc/reprints?id=1-28ZA4X2M&ct=220202&st=sb. Accessed: 14 March 2023.

OpenText 2023. Introducing opentext.ai and OpenText Aviator. URL: https://investors.opentext.com/press-releases/press-releases-details/2023/Introducing-opentext.ai-and-OpenText-Aviator/default.aspx. Accessed: 14 August 2023.

Research and Markets, 2023. Customer Communication Management (CCM): Global Strategic Business Report. URL: https://www.researchandmarkets.com/reports/5141052/customer-communication-management-ccm-global. Accessed: 31 August 2023.

Rouhiainen, L. 2020. ARTIFICIAL INTELLIGENCE 101 Things You Must Know Today About Our Future. E-book. Accessed: 14 September 2023.

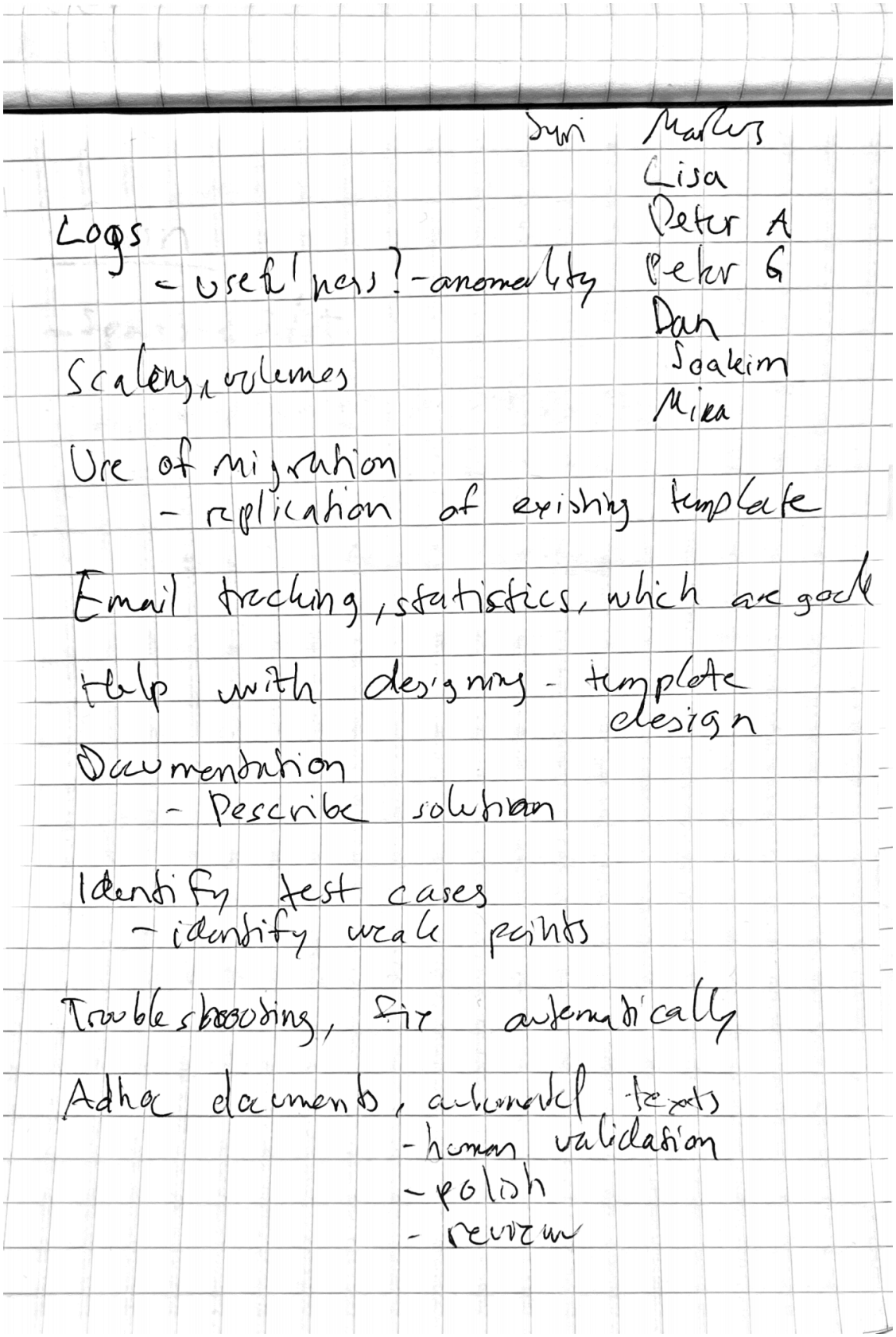
Taylor, P. 2023. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025. Statista. Accessed: 10 September 2023.

Thormundsson, B. 2023. Artificial Intelligence market size 2030. Statista. Accessed: 14 March 2023.

Yang, G. 2006. The History of Artificial Intelligence. University of Washington. Washington. URL: https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf. Accessed: 27 October 2023.

Appendices

Appendix 1. Brainstorming session notes



Appendix 2. log-classifier.rst

```
.. code:: ipython3

    ### ENIT log classifier tryout ###

    ### uncomment to install below dependencies if needed
    #!pip install -U pandas
    #!pip install -U scikit-learn
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn import svm

    # read the file and print sample
    samples = pd.read_csv('labels.txt', sep=';')
    samples

.. code:: ipython3

    # convert to lowercase and print out sample
    samples['LogText'] = samples['LogText'].str.lower()
    samples

.. code:: ipython3

    # create training set and testing set
    z = samples['LogText']
    y = samples["Label"]
    z_train, z_test,y_train, y_test = train_test_split(z,y,test_size =
0.2) # test_size is percent of test cases from total samples
    print(z_test, y_test)
```

```
.. code:: ipython3


    # Exstract features

    cv = CountVectorizer() # transforms words into random numbers and
counts their occurencies

    features = cv.fit_transform(z_train) # fits the training set into
vectorized values

    print(features)


.. code:: ipython3


    # using SVC (Support Vector Classifier) model

    model = svm.SVC()

    model.fit(features,y_train)


    # test the model

    features_test = cv.transform(z_test)

    print('Accuracy score:', model.score(features_test,y_test))


.. code:: ipython3


    import joblib

    # dump models into binary to be called by the webservice

    joblib.dump(model, 'trained_model.pkl')

    joblib.dump(cv, 'vectorizer.pkl')```
```

Appendix 3. log-classifier-web-service.py

```
from flask import Flask, request, jsonify

import joblib


app = Flask(__name__)

# Load the trained model and vectorizer

model = joblib.load('trained_model.pkl')

vectorizer = joblib.load('vectorizer.pkl')


@app.route('/predict', methods=['GET'])

def predict():

    input_string = request.data.decode('utf-8')  # Decode the received
data as a string


    # Preprocess and vectorize the input string

    input_vectorized = vectorizer.transform([input_string])


    # Make a prediction

    prediction = model.predict(input_vectorized)[0]

    response = {'classification': prediction}

    return jsonify(response)


if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5001, debug=True)
```