Utrecht University

BACHELOR THESIS

Automated medical transcription de-identification pipeline

Author:
Jacco J.S. Broeren

Supervisor:
Prof.dr. Sjaak Brinkkemper

Second Examiner:
Dr. Fabiano Dalpiaz

Department of Information and Computer Sciences

September 26, 2023

Abstract

Department of Information and Computer Sciences

Nowadays, privacy within the Information Technology domain is a hot subject. Privacy concerns arise with the development of increasingly smart automated summarisation and interpretation systems which use Natural Language Processing (NLP) technologies. Just like the Care2Report system, which is currently under development at the Utrecht University. This system uses transcriptions in order to automate Electronic Patient Dossier entries made after doctor consultations. This research aims to tackle a privacy concern found in this system by designing a pipeline which de-identifies named entities (names) found in medical transcription texts before these are processed by the Care2Report system. The created treatment is able to de-identify named entities by replacing names found in sentences of different languages, in real-time and on a system with limited processing power. Therefore, this research concludes that it is possible to create such a de-identification pipeline. However, the research also finds that many parts of the system should be optimised further in order to completely solve privacy issues.

by *Jacco J.S. Broeren*

Contents

List of Figures

List of Tables

Acronyms

ASR Automated Speech Recoginition.

ATENE Automatic Transcription Evaluation for Named Entity.

C2R Care2Report.

CPU Core Processing Unit.

GDPR General Data Protection Regulation.

GPT Generative Pre-trained Transformer.

HIPAA Health Insurance Portability and Accountability Act.

LIG Levenshtein-ISG-Guth.

MHR Medical Health Records.

ML Machine Learning.

NEM Named Entity Matching.

NER Named Entity Recognition.

NEWER Named Entity Word Error Rate.

NLD Natural Language Detection.

NLP Natural Language Processing.

NMA Name Matching Algorithms.

PHI Personal Health Information.

Chapter 1

Introduction

1.1 Research Aim & Relevance

The current Care2Report system processes identifiable data which is sourced from doctors appointments and other conversations. In order improve the privacy of the systems users, a method has to be found which can de-identify transcription data before it is processed by the Care2Report system. The aim of this research is to find such a method using existing literature.

1.2 Overview

This thesis discusses the process of creating a de-identification pipeline using design science principles defined by Wieringa: Problem Investigation, Treatment Design, and Treatment Evaluation. It aims to find out whether it is possible to create a pipeline which de-identifies real-world entities in a sentence by replacing their name with a different name. Other identifiers such as locations or physical identifiers are not within the scope of this research. In the first section, the research method is presented including a further elaboration of the Design cycle. This is followed by the problem definition. In this section, the de-identification pipeline is first discussed. The different steps that are performed within the pipeline are extracted from previous work and combined. Furthermore, each step in the pipeline is discussed separately. The technologies are elucidated and an exploration to find optimal solutions for each step is conducted. Subsequently, a treatment for the previously presented problem is designed. Also, using the information gathered in the problem definition, requirements are created and tested. Then, the treatment is evaluated; previous requirements are used to test the pipeline. Lastly, the discussion and conclusion of this thesis are presented.

1.3 Research Questions

MRQ: **"How can a pipeline be constructed to de-identify medical transcriptions, which can be used with the Care2Report system?"**

RQ1: "What steps need to be performed in order to construct a pipeline to automatically de-identify transcriptions?"

RQ2: "How can entities be detected in perfect transcription data?"

RQ3: "How can the de-identification process be performed in real-time?"

RQ4: "Can the de-identification pipeline perform in real-world simulations?"

Chapter 2

Method

The method used to distil and solve the posed problem is sourced from [27]. This research method is used to treat a technical problem within the information technology domain, using scientific research methods.

2.1 Wieringa Design Cycle

The Wieringa Design Cycle aims to solve a technical research problem by iterating over three steps which are part of the Design Cycle (Figure 2.1). In each step information is gathered and analysed.

2.1.1 Problem Investigation

During the problem investigation general data about the context of the problem is gathered. This includes the involved stakeholders and their goals. Also previous research and existent treatments for similar problems are used. This information is used to create a high level overview of all the previously available data and the problem that needs to be treated during the iterations. Combining the stakeholders and their goals, together with previous research leads to a clear problem definition and a theoretical framework.

2.1.2 Treatment Design

Using the stakeholder goals and previous research, requirements are engineered. Already existent treatments to different *challenges* that were discovered in the previous step are combined. This results in a treatment for the technical research problem.

2.1.3 Treatment Validation

In this section the treatment that was designed will be tested. This is done according to the described requirements. A requirement contains one or more conditions which need to be met in order to be considered successfully treated. When all requirements are met, the treatment is considered a successful treatment to the earlier described technical research problem.

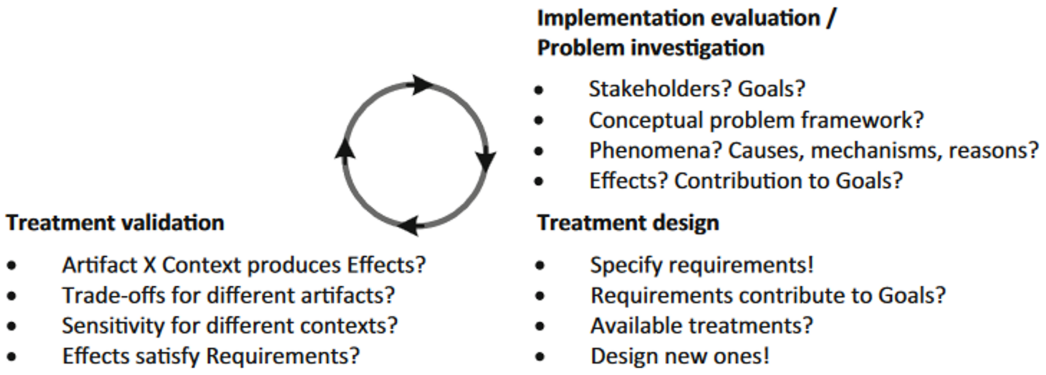

Figure 2.1: Wieringa Design Cycle

Chapter 3

Problem Definition

This chapter will discuss the problem that is treated in this thesis. This is done using four sections.

1. The problem context.
2. Social context.
3. Knowledge Context.
4. Problem framework & definition.

The problem context The reader is provided with a description of the context in which this problem is found. An insight into the architecture and use of the Care2Report (C2R) system is given. It is the foundation of the research as the treatment should be implemented within this already existent context.

The social context Involved stakeholders and their goals are presented next. These are considered as they provide a more clear focus for the research as well as limitations to the possible treatments. This information is also needed later on in order to engineer the requirements of the treatment.

The knowledge context All related work and research used to compile the pipeline and find treatments for every step of the pipeline is presented in this section. This is done by taking the reader through a high level overview of the treatment first. Existing solutions to closely related problems are used to create a list of steps which are performed in order to treat the problem which is uncovered. After this high level overview, each step is discussed separately. For each step, an optimal treatment has to be found. This is done using different sources or proposed solutions and comparing them. All this information can then be used to design the optimal treatment for this problem. This is done in the next chapter (Chapter 4).

Problem framework & definition The chapter concludes with two high level tools that are used in the Wieringa design cycle. The conceptual problem framework and technical research problem. These two pieces of information aid the scope and direction of the research as it helps to clearly state the essence of the problem that needs to be treated.

3.1 Problem context

The Care2Report process

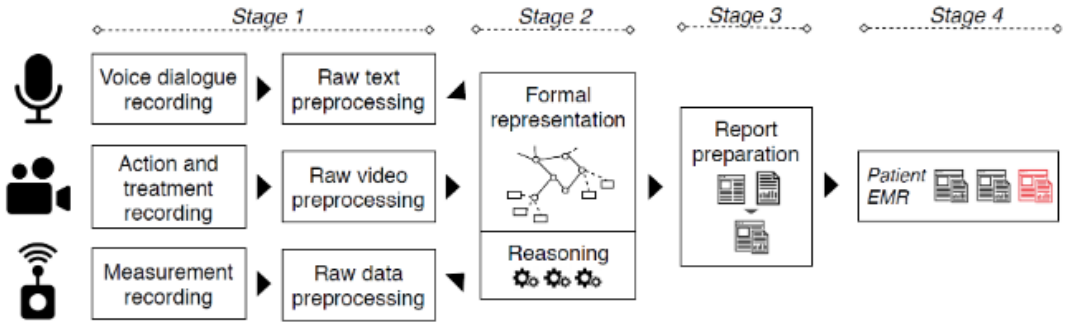

Figure 3.1: Care2Report process

The C2R system aims to automate the entry of consultation data into the Electronic Patient Dossier system. During the consultation different types of data are gathered. Audio and video data for instance. Other data, sourced from scales, blood pressure sensors, thermometers etc. is also used. This data is sent to a server where the data is processed. For this process, AI (Artificial Intelligence) and Rule-Based systems are used. The output of these systems are used to generate a report. This report can then be checked by the doctor that performed the conversation with the patient. Small adjustments can be made, and the report can be filed into the Electronic Patient Dossier system. The system aims to minimise the administrative load on doctors by automating the filing of consultation reports.

The Care2Report System

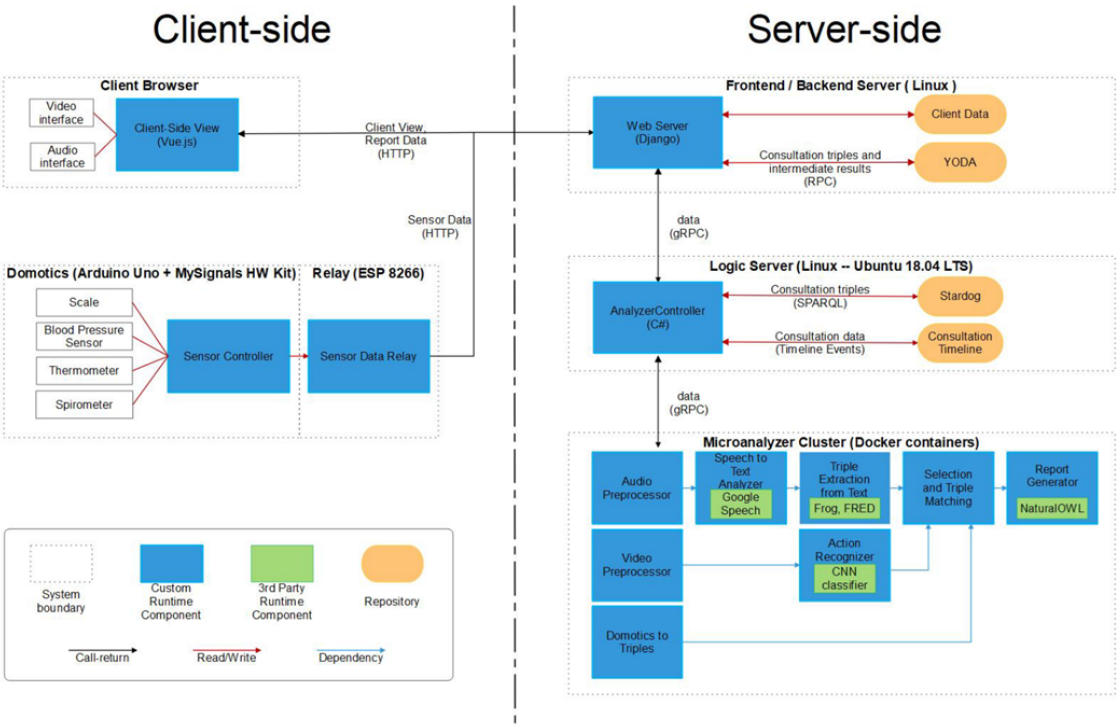

Figure 3.2: Care2Report system architecture

This process takes place inside a client-server architecture. This architecture was chosen because it allows computationally heavy tasks to be executed on a dedicated server and only let the client handle light-weight pre-processing tasks and interaction with the system. The de-identification of data is also a computationally heavy task as it will involve several algorithms and Machine Learning (ML) models. These tasks will be performed server-side. The treatment of this thesis will be situated in an extension of the current server-side of the system. The current system works with docker. This architecture can also be used for this pipeline. The system would be situated inside one of the docker containers and accept input data through an API endpoint which is accessible through the docker container. Therefore closely resembling the current data-streams and fitting in with the current setup.

3.2 Social context

The stakeholders which are involved with the creation of this de-identification pipeline are depicted below. Each stakeholder also has their own goals which are met with the creation of this pipeline.

- Care2Report R&D team
  - Implementation of a de-identification pipeline.
- End-users:
  - De-identification of personal data to ensure privacy throughout the system.
  - Perfect re-identification to ensure perfect system output.
  - Prevent loss of required information to allow correct processing by C2R.
- Developers
  - Easy deployment
  - Accessible source-code.
- (Medical) Research Community
  - Output from the pipeline can be used for the training of other models
  - De-identified transcripts can be used to train new doctors on real-world examples that are not linked to any person directly.

3.2.1 Care2Report R&D team

The R&D team has the need to get such a de-identification pipeline in order to further improve currently existent parts of the system.

3.2.2 End-Users

End-users are those whose data is processed by initially the C2R system, and therefore the de-identification pipeline. Their concerns are centered around improving privacy by removing identifiers from the to-be-processed texts. For these stakeholders it is also vital that the de-identification pipeline will not interfere with the quality of the output from the C2R system. De-identification needs to be done in such a way that the system still can process the data and no important information is lost.

3.2.3 Developers

Developers working on the C2R system are also stakeholders of this project. As they need to work on the full system, the de-identification pipeline is also included. In a report from the C2R project named *"Report: Care2Report Easy Installation"* the installation process is mentioned including several requirements concerning the development and deployment process. In order to hold on to the deployment and development principles of the C2R project in this project as well, the goal of the developer is that this pipeline can be deployed locally and singularly. Without the need of deploying other parts of the system as well. It needs to be *containerized*.

3.2.4 (Medical) Research Community

Indirect stakeholders of this project are external researchers. When a sufficient degree of de-identification is reached (Anonimization), it is possible to release raw anonimised transcription data to the research community. Research within the medical domain or Natural Language Processing domain for instance.

3.3 Knowledge Context

3.3.1 The pipeline

De-identification of medical health records has been researched extensively. Due to the high demand of publicly released medical health records and its short supply, automated de-identification is one of the go-to technologies to de-identify structured medical patient data. This data is needed for the development of applications like Natural Language Processing (NLP) and Machine Learning (ML). De-identification of transcription data sourced from doctor consults specifically has not been researched previously. Previous research ([6] [18] [24] [26]) regarding the de-identification of Personal Health Information (PHI) all point the same two basic steps to de-identify medical personal data:

1. Named Entity Recognition (NER)
2. Removal or replacement of these entities.

This research also points to a set of 17 entity types that should be detected and removed in order to attain a sufficient level of de-identification to call it anonimisation. These are entities which are deemed PHI by the Health Insurance Portability and Accountability Act (HIPAA). General Data Protection Regulation (GDPR) also deems these datatypes to bo protected. HIPAA and GDPR are laws regarding online safety. Both should be considered as HIPAA is leading in US markets and GDPR is leading in European markets.

1. Names
2. Geographic locators
3. Elements of dates
4. Telephone, cellphone and taxnumbers
5. Email addresses
6. IP addresses
7. Social Security Numbers
8. Medical Record Numbers
9. Health plan beneficiary numbers
10. Device identifiers and serial numbers
11. Certificate / license numbers
12. Account numbers
13. Vehicle identifiers and serial numbers, including license plates
14. Website urls
15. Full face photos and comparable images
16. Biometric identifiers
17. Any unique identifying numbers, characteristics or codes

Almost all of these datatypes (images, IP addresses, account numbers, Device identifiers, certificate or license number, website URLs and bio-metric identifiers) will not occur in the input data of the de-identification pipeline and will therefore be ignored. Data that does appear in the input, but cannot be replaced as it would impact the C2R pipeline are *names of days, months and years*, *numbers* and *geographical locators*. Names of days, months and years need to remain in the pipeline output as it could contain important information about medicine a patient has already taken before the consultation. For instance, the patient comes in with a tummy ache that started on Wednesday and mentions they already took 50mg of paracetamol. The information about the amount of medicine a patient has taken can impact the further processing by the C2R system. The same goes for the day at which the first symptoms occurred. A way to get around this issue and replace only the information that is going to be used by the C2R system, is to analyse the context in which

the information is given by the patient or the doctor. This was done in [18]. However, this would introduce another step of context interpretation into the pipeline and make the de-identification pipeline even more complex. This will therefore fall outside of the scope of this research and not be considered. Therefore days, months, years and other numerical values that occur in the pipelines input will not be replaced. The pipeline will only replace names of people.

Going back to the previously described technical research problem. The pipeline needs to work with different languages. This adds an extra step to the two previously mentioned steps. A technology known as Natural Language Detection (NLD) can be used to identify the language of natural language textual data.

When the natural language is identified and Named Entity Recognition has been performed, entities need to be de-identified. This will be done using a replacement process. Names are changed to different names. Not all named entities should be replaced. Within the medical domain, diseases are sometimes named after a person. This creates a challenge. NER could detect a disease as a person. Also, variations in grammar could meddle with the replacement and entity matching. The combination of the problems found in this task of entity replacement introduces two extra steps: Eponymous disease detection and Named Entity Matching (NEM).

The re-identification step also needs to be added to the list of steps. Because this pipeline is designed to be used with the Care2Report system, re-identification needs to be performed after the data has been processed by the system. This data can then be sent to the doctor for verification and is submitted into several output systems of Care2Report (*figure 5.2*).

This leaves the research with a clear list of steps to create and combine into a pipeline:

1. Natural Language Detection (NLD)
2. Named Entity Recognition (NER)
3. Named Entity Matching (NEM)
4. Eponymous disease detection
5. De-identification

3.3.2 Natural Language Detection (NLD)

NER models are natural language specific. To overcome this problem, language specific NER models are trained and implemented. In order to apply different models to sentences of a different language, Natural Language Detection (NLD) is needed. This subject has been researched extensively. There are numerous tools available and the overall quality of these tools is very high [13]. The tools that are available are either trained on speech transcription data or textual data. A highly cited model according to *Google Scholar* is LangId.py [13]. An off-the-shelf Natural Language Detection tool for Python. This model has been trained on a data set which is sourced from five different domains containing 97 languages [12]. langid.py is compared to other alternatives showing its competitiveness and completeness in relation to the other available tools such as LangDetect [20], TextCat [25] and CLD [15], it is concluded that Langid.py is state-of-the-art and useable with a wide variety of texts, including transcription data [13]. The tool also allows to limit the amount of languages that are detected resulting in a much higher accuracy as the country and possible languages can be set in advance or be extracted from the available information about the context of the recorded conversation.

3.3.3 Named Entity Recognition (NER)

NER (Named Entity Recognition) are *natural language processing model*s used to detect *real world entities* within natural language such as people, organisations, locations or numerical values. Natural language can be provided in both audio or text format. The training process of a NER model usually entails a training pipeline using the following steps [24]:

1. Manual or Semi-Manual entity classification on data originating from the domain that the model is going to be applied on (e.g. Medical, Legal, Governmental, Consumer, etc.).
2. Creation of Training and Testing set containing annotated data.
3. Training the model using the training set within either a completely blank model or a pre-trained model. Fine-tuning an existent model is also possible.
4. Evaluate the model quality using one or multiple NER specific benchmarks such as Named Entity Word Error Rate (NEWER) [8] or Automatic Transcription Evaluation for Named Entity (ATENE) for an ASR-NER pipeline [10]. Along with standardised benchmarking such as F-score, Accuracy, Recall and Precision [1].
5. Reiteration of the training and evaluation steps or proceed to model release.

The input data of our NER model can be in both speech or text format. Depending on the type of input data, a choice can be made. It is possible to *separate the Automated Speech Recoginition (ASR) and NER task*. First, the audio is transcribed. Thereafter, the result is fed into the trained Named Entity Recognition (NER) model [9], [29]. A different approach is to *combine the ASR and NER into a single model*, as was done in [11], [9] and [29]. There it was found that, when combining the tasks, higher model performance can be attained. This is because research suggests that ASR quality can be improved using NER to correct or detect errors made by the ASR task. Combined ASR-NER models are able to detect wrongly transcribed words and can intercommunicate the mistakes between the ASR and NER model to review some of the transcription data. The model can ignore or improve that particular entity or the sentence itself. However, using this approach within the projects current domain is unfeasible. To train a model that is use-able in the medical ASR/NER context, audio recordings from actual consults are needed. These are very scarcely available. This points us to the initial approach.

However, separate NER and ASR models do not intercommunicate transcription errors. Even worse, errors made by the ASR model are carried into the NER model. NER models used on audio transcriptions are often trained using perfect transcription data. This causes the model to fail when handling imperfect transcription data, which is the reality. The other way around, models that were trained on imperfect transcription data also have recognition errors incorporated into the model. Correct sentences can be perceived as incorrect which could cause entities within that sentence to be falsely ignored or detected. Recent research suggests that training the NER model on the output of the implemented ASR model decreases NEWER [23]. However, this also makes them highly coupled, which is undesirable as it creates a dependency between different parts of the pipline and goes against the code principle of *low coupling*.

NER models are available in a public repository to dynamically load and use NER models. This repository is known as *Huggingface* [28]. This repository provides a solution that allows to hot-swap models by importing and deploying them from an external source on the go. It support dynamical model embedding, which enables the pipeline to be programmed dynamically instead of hard-coded around a particular model (*low coupling*). Because model quality is still improving a lot in short periods of time, it is best to support quick-swapping of NER models. With a multilingual

pipeline, maintainability also improves when community sourced models for different languages can be incorporated into the pipeline. Community models are often maintained by external parties, eliminating the need to maintain or update the used models for this specific pipeline. It is also possible to upload models to this repository, which would make it possible to use internally developed models without the need of altering the pipeline itself. Summarising, it provides a future proof solution that incorporates many possible development scenarios.

A different possibility is presented through the emersion of the third generation Generative Pre-trained Transformer (GPT) model. OpenAI has produced three GPT models over the past years. These models were trained unsupervised making them applicable for a far broader spectrum of tasks (and languages) then supervised trained models. Unsupervised models are fed with a large amount of information, in this case texts sourced from a lot of different online sources such as Wikipedia, blogs, GitHub, archives, news websites and so on. These models then train themselves by finding patterns in this dataset. Therefore the model is not trained on a single task making them usable for a lot of different tasks. Since the input language was also varied, the model is also able to work with different languages. Regarding the application of NLP by the GPT. The model is initially fed a number of large data sets. This data is incorporated into the model. In order to then use this model to perform a specific task such as NER, the model is fed with a small amount of prompting data. This will fine-tune the model to perform a specific task using the knowledge it gained through initially processing the large dataset.

The model from OpenAI, of which the paper [17] was released in 2018 performed well with very little prompting, which was an improvement of the state-of-the-art. The model needed only three epochs to complete most downstream tasks, meaning minimal fine-tuning was needed to accomplish sufficient results on various tasks. Its zero-shot performance on these tasks was also decent, this was a significant improvement compared to other state-of-the-art models back then. GPT-1 was the first model produced by OpenAI [17]. What followed was GPT-2. The GPT-2 model was far larger having 10 times more parameters then the GPT-1 model. Its vocabulary was also much bigger and it could process more text. The GPT-2 model performed a lot better on the NER task. It increased the state-of-the-art accuracy for this task with 7% [19]. Then, GPT-3 came around. This model has 100 times more parameters then its predecessor and performs even better on downstream tasks with few- or zero-short learning. Due to the huge amount of data that was captured during the training phase, this model even developed capabilities like writing human-like texts and code using natural language descriptions. It is also able to perform tasks in multiple languages although performance varies depending on the language used. The models main benchmarks have been done using the English language which is where its most capable. OpenAI initially provided source code to their models, but with Microsoft acquiring the organisation, the source code of their most recent model has been withheld from the public and only been made available through their API. The main argument to do this, was that this model could potentially threaten online communities. It could impose as a person or write fake news articles within seconds. Their API puts heavy financial strain on using this model for tasks that require large streams of data to be processed. Whether using the OpenAI model is financially feasible remains a question to be explored.

3.3.4 Named Entity Matching (NEM)

Names are identifiers for real-world entities. The problem with names extracted from sound is that phonetically identical names can have spelling variations, or textually identical names can be pro-

[0]Zero-shot learning describes the ability of a model to perform a task without the need of seeing any examples during earlier training stages in order to understand the specific task.

nounced differently. When placeholders are applied to named entities in order to de-identify them. These placeholders should be identical for each real-world entity, so that further processing by the C2R pipeline is successful[1].
To illustrate the issue: The ASR model might transcribe the phonetically written name *Tam* as both Tom and Thom in two separate sentences. The NER model will recognise both names as a person and store them in a central database. Thom is replaced with one of n placeholders. Tom is also detected in another sentence, but since that name is textually different then Thom, the script replaces this name with a different placeholder. During further processing these two people are not matched by the C2R system which causes it to not fully detect all client related information that was captured in the transcribed consult.

In literature, algorithms that are able to label these phonetically or textually identical names as a single world entities, are known as Name Matching Algorithms (NMA) [21]. The current use-cases for these algorithms are found in the legal domain: Transcribers might denote names differently causing a single legal entity to be denoted differently [5]. This requires the use of NMA to track these entities and link them where possible. A comparative study from Snae (2007) [21] showed that there are several ways to match named entities. In this paper three types of variations in names were reviewed; Character variations, Phonetic variations and Spelling variations. Using different types of algorithms, these variations can detected and filtered automatically. According to [21] these four types of algorithms are:

1. Spelling or string based algorithms.
2. Phonetic or sound based algorithms.
3. Composite algorithms (a combination of multiple algorithms of type 1 or 2).
4. Hybrid algorithms (combining 1 and 2).

The paper found that hybrid algorithms are the best solution for the Named Entity Matching task in general. They note however, that the choice of name-matching methods depends on the specific application of the algorithm. Our variations will be spelling based. Character variation describes the differences between entities being denoted using uppercase, lowercase or capitalised names. Typically, ASR output is non punctuated, non capitalised. Which removes Character variation from the variation list. Phonetic and spelling variations are left. Since users could pronounce a name differently then is meant, this type of variations are to be considered as occurring. Spelling variations too, as the ASR model could accidentally denote a name with the same pronunciation using different spellings. According to [21] these two variations are best handled by Levenshtein-ISG-Guth (LIG) hybrid similarity measures. They are well equipped to handle multi-cultural names which is also important for our pipeline as it will process multi-lingual data and names which find their origin in different cultures.

The LIG similarity measure was proposed by [22]. The third of three variations of the initial LIG similarity measure that was proposed is the LIG3 measurement. In this measurement, the words are mathematically processed and a value is computed which represents the similarity between these names.

3.3.5 Eponymous disease detection

Not all entities that are detected using NER should be de-identified. Eponymous diseases, which also mentioned within the medical de-identification domain by [26], are diseases named using names

[1] To limit the amount of research variables, it is assumed that n people in each transcription are denoted by different names. So no two people with an identical name will exist within our transcript

that are also commonly used to identify a person. In [26] these entities are described as non-Personal Health Information (PHI) entities. The systems used to de-identify Medical Health Records (MHR) generally perform well on so-called ambiguous PHI data[2] according to [26]. In this research it was found that models employing some form of ML perform better on recognising the difference between PHI (names) and non-PHI (eponymous disease) data when de-identifying text. Rule-based systems perform generally worse. Currently, the are no models available to de-identify medical transcription data. Therefore, this project is left with rule based systems.

Rule-base systems need a database to search for these terms. SNOMED-CT is a multilingual, highly regarded, very complete database containing both diseases and medical terminology.

3.3.6 De-identification of entities

Classified entities are replaced in the input data by placeholders. The type of placeholder that is chosen impacts the processing of the data during later stages of the Care2Report pipeline. There are four options:

1. Randomised real-world alternative [26].
2. Denote the client and caregiver always with the same real-world placeholders[3].
3. Denote the client and caregiver as *client* and. *caregiver*[4].
4. Randomised string value.

Each of these solutions have an influence on the further model performance inside the C2R pipeline. The current C2R speech processing pipeline uses unprocessed text as input. Therefore, to resemble the initially existent process, choosing randomised real-world alternatives would be closest to the already existing process (option 1). However, it could be beneficial for the pipeline to take a set of n standard placeholders (option 2) that are filtered for other medical domain uses (Eponym diseases, domain terminology, etc.) and could therefore enhance the performance of down the line NLP processing tasks. Option 3 could pose as an interesting alternative as well. It could even further enhance down-the-line processing by already taking the people inside the textual data and feeding that explicitly to the processing task. But, it should be noted that this requires the NER task to be executed perfectly. Also modification to incorporate this data in the C2R process should be made. It will create a higher degree of coupling between the different parts of the system which is not desirable.

3.3.7 Re-identification of entities

The entities that were replaced by the pipeline need to be re-identified in the C2R output. This can be done by going over the text while searching for the used placeholders and replacing those placeholders to its original entity again. The data retention of the C2R system is none [14]. Meaning all data that is processed by the system for each consultation is removed from the database at the end of the processing session. How this data retention is handled by the de-identification pipeline needs to be investigated.

[2] Eponym diseases or other terminology originating from the medical domain
[3] Data about the Client and Caregiver are needed during de-identification in order for this possibility to be allowed
[4] Variation on [18], denotation there was done using PERSON like placeholders.

3.4 Conceptual problem framework

The conceptual framework for this study is depicted in figure 3.4

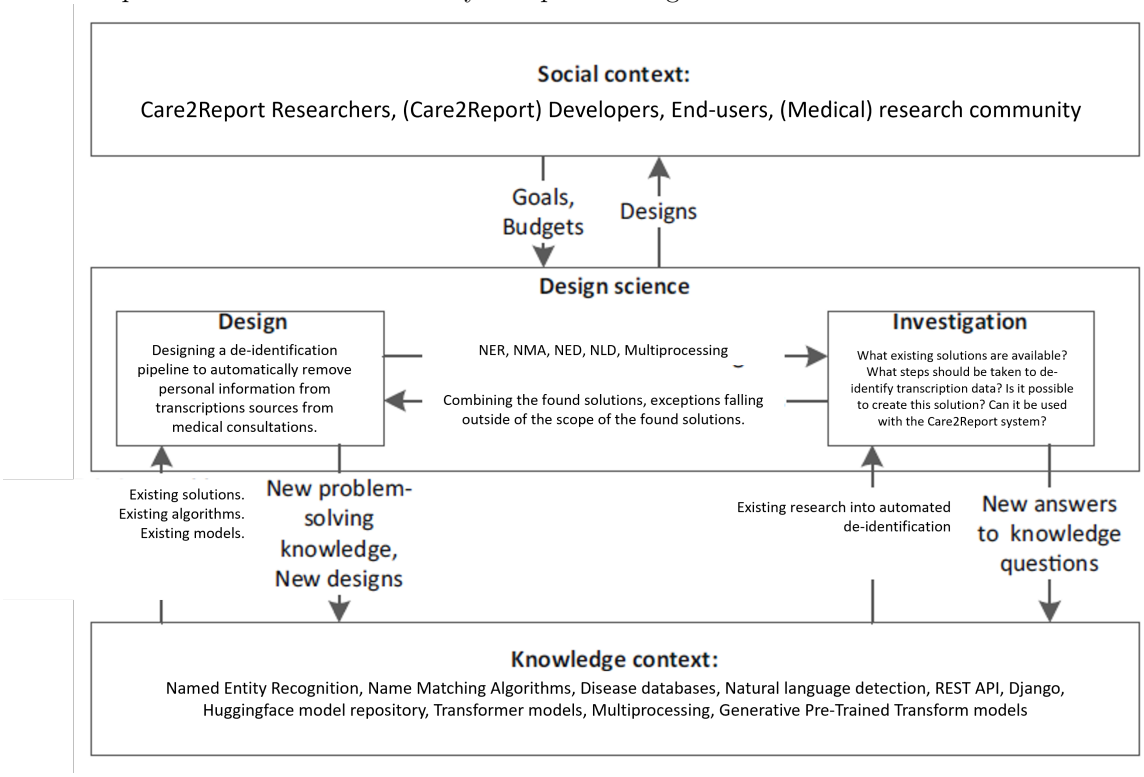

Figure 3.3: Conceptual Framework

Using the data presented above, the following technical research problem can be compiled:

> *"**How to** create a de-identification pipeline **that** works with different languages, in real-time and is lightweight **so that** the names of end-users are replaced with real-world identifiers to improve their privacy **in** the Care2Report pipeline?"*

Chapter 4

Treatment Design

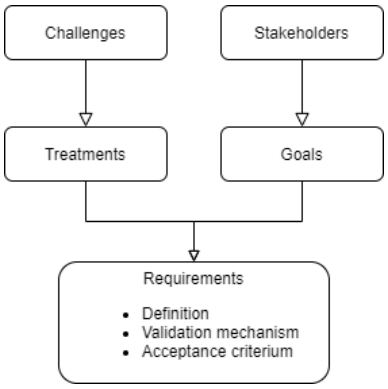

Figure 4.1: Requirement derivation process

To successfully design the treatment, the problems from chapter 3 need to be treated and its treatment should be tested. To do this, challenges are extracted from the problem definition. These challenges are treated separately and together will lead to a complete treatment. Therefore, each subsection covers one of the pipeline steps and its challenges. A challenge is defined and placed in context of the pipeline, after which the treatment is presented. To test the combination of treatments, requirements are engineered. In order to limit the scope that is covered by requirements to test the successfullness of the pipeline, a number of assumptions are made. These assumptions are also presented in this chapter. Each requirement consists of three parts:

1. Definition
2. Validation mechanism
3. Acceptance criterion

The problem that is treated in this chapter was defined in the previous chapter. The treatment design is the solution to the technical research problem which was the following:

> *"**How to** create a de-identification pipeline **that** works with different languages, in real-time and is lightweight **so that** the names of end-users are replaced with real-world identifiers to improve their privacy **in** the Care2Report pipeline?"*

4.1 Challenges

The challenges presented below are extracted from the information given in chapter 3. A recap is given together with a definition of the challenge and the treatment that needs to be applied in order to tackle the challenge.

Named Entity Recognition

The data that is fed into the pipeline is processed by a NER model that is trained using mostly written, single language texts from a specific domain. Because the model quality (recall and accuracy) is highest when the model is trained using domain specific data and most training data that is available is only sourced from written sources. A challenge occurs when the desired solution is a pipeline that can process multilingual speech (transcriptions) from the medical domain, or even more specific, medical consultations. Processing of data by a model can also be computationally expensive when models are large. Furthermore, current state-of-the-art models are quickly outdated. It should be considered that changing to better or improved models needs to be possible with as little down-time as possible. The above mentioned problems should be tackled individually. They are all part of the NER step in the pipeline.

Multilingual - Multi language input is problematic as models are far more accurate when they are language specific. It allows the model to be trained on more specific data which results in a higher precision. Therefore a treatment has to be found to use different NER models for different languages. The treatment to this challenge lies with the dynamic use of different NER models depending on the natural language of the text. This is done through the implementation of Natural Language Detection (NLD) which was discussed in 3.3.2. The proposed package, LangID is also the package that is used to treat the problem of multiple input languages.

Multiple steps in the pipeline need to be able to handle different languages.

- The NER model is different depending on the language that is fed into the pipeline.
- The SNOMED-CT release that is queried to find possible eponymous diseases is different depending on the language.

Dynamic use of models - Since models are outdated quickly and require frequent updating, a treatment has to be found for the dynamic use of different models while the pipeline is already deployed. Models need to be instantly interchange- and update-able. A treatment for this problem is the use of a model repository. Such a repository exists in the form of a large public model database named "Huggingface". This database contains numerous state-of-the-art NLP models. New and improved models are also added continuously. When a model is swapped with an already implemented model, the new model is downloaded, loaded into the processing memory and immediately useable upon launch. Model updates are also handled dynamically, they are automatically deployed after they are downloaded. Changing to a different model is also possible. Swapping to a different model

is as simple as changing the reference to the repository in the script. Adding different languages can be done in the same way. One can add the language to the dictionary containing the languages, add the model reference to Huggingface together with the SNOMED-CT edition from that language. The model is then retrieved upon first use and loaded together with the already existent models.

Named Entity Matching

Entity classification for correct replacement also provides a challenge. Entities can be denoted differently while they represent the same real-world entity as was described in 3.3.4.

Entity linking - Real-world entities can be written down using different words. Variations in spelling or capitalisation can occur. To make sure the replacement process for each real-world entities is consistent, a treatment has to be found to link all differently denoted real-world entities. The use of NMA will treat this issue. By matching and combining entities, correct entity replacement can be ensured.

The treatment used is the LIG3 algorithm. This algorithm calculates the probability of two names which are spelled slightly different to be actually the same [22]. It is a 'Levenshtein-ISG-Guth hybrid similarity measure'. Which is mathematically defined as:

$$sim_{LIG3}(X, Y) = \frac{2I}{2I + C}$$

In this equation, I denotes the number of matches between words A and B, which is then truncated to the length of the shortest word. C is the Levenshtein distance between the two words. The threshold at which two names are considered the same still has to be found and implemented. Optimising this threshold to a perfect precision does not fall in the scope of this research. The LIG3 algorithm can be implemented in a python script with the use of the Abydos package [3].

Eponymous disease detection

Disease database - Some diseases are named after a person, which the NER model will detect as an entity. To treat this issue, a database containing eponymous diseases should be queried for each detected entity. When an entity is found in this database, the treatment should refrain from replacing that entity with a placeholder. This way, the C2R input will never be compromised. A treatment that can be used is SNOMED-CT. A database with all diseases. When the pipeline has a hit on one of the entities, that particular entity is not replaced by the pipeline.

De-identification

Placeholder selection - In order to successfully replace existing entities in the transcriptions, the placeholders should not be used in the medical domain. However, the choice of placeholders can be broad as the current system works with raw data. Therefore the choice of placeholder can be standardised to a common name representing said entity. By using a list of predetermined placeholders with a sufficient length where no danger of running out of names is present, this challenge can be treated. The list that is used within the treatment can be found in Appendix A.6. These names are commonly used, gender neutral names. Since the NER algorithm does not know whether a name is male or female, the choice was made to go with gender neutral names. The NER algorithm also does not know whether a first or last name is used. This is also ignored in the replacement process. All named entities are replaced by a gender neutral first name.

Realtime processing

Multiprocessing - To allow for real-time processing of text, the designed pipeline needs to be able to concurrently process multiple separate sentences. General scripts run on a single core. All processing is handled by a single core and processes are ran one after another. Asynchronous processing makes it possible to utilise multiple cores at the same time, performing multiple tasks simultaneously. Multiprocessing is a native Python module which enables coders to run code asynchronously [2]. This module utilises all cores on a CPU.

Lightweight

Knowledge Distillation - With the use of pre-trained models. No on-site training needs to take place which makes the application much more lightweight to run. Also depending on local requirements concerning model accuracy and size, the decision about the implemented model can be made locally on a per language basis which leaves room to customise the computational needs by the server significantly. Current trends in the NLP model training are shifting away from multi-billion parameter models, and moving towards smaller more efficient models [7]. This practice is known as "knowledge distillation". It means *"making the model smaller by taking a large model that works very well. And distilling it into a smaller model, which are then creating the larger models' predictions"*. The actually needed parameters are retained and the unnecessary knowledge contained in the model is discarded. Incorporating the easy interchanging of language models will allow for this application to move with this trend. The creation of smaller, as accurate, models. Therefore resulting in a more future proof lightweight application.

Fit within the Care2Report system

Architecture - In order to create a treatment that fits into the Care2Report system, the designed treatment needs to be usable from the current Care2Report system with as little changes as possible. This is achieved by using a REST API as entry point for the de-identification pipeline. The de-identification pipeline can be deployed next to the already existent system inside a docker container and then be reached for de-identification through the API endpoint. If the Care2Report team chooses to fully incorporate the pipeline into the system, they can also choose to merge the code of the de-identification pipeline directly into the Care2Report system. This is possible as the de-identification process can be deployed on a per-sentence basis with only three lines of code.

```
process = de_identify(textdata, session)
process.daemon = True
process.start()
```

The text-data object consists of a python dictionary containing information about the database entry of the sentence. And the session object consists of a python dictionary containing information about the ongoing transcription session of which the sentence is a part of.

The web-server used to interact with the de-identification pipeline is built with the python based Django framework. This is the same python web-server as was used in the original Care2Report system. This choice was made to align the technologies used in order to allow the Care2Report development team to easily work with the newly created treatment.

4.2 Treatment

Using all the information given in the previous section, a treatment was designed. The treatment is a pipeline *that works with different languages, in real-time and is lightweight.* The pipeline can be found in figure 4.2. The pipeline will be placed inside the audio pipeline which can be found in figure 3.2. There it is situated as a "de-identification" step between "Speech to Text Analyzer" and "Triple extraction from Text".

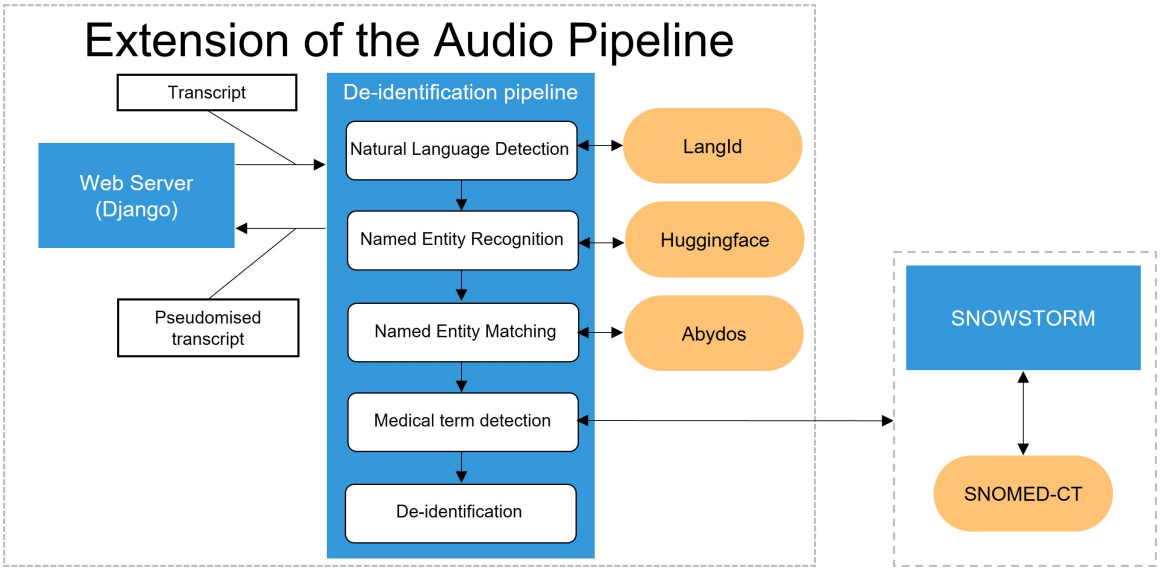

Figure 4.2: De-identification pipeline.

NLD is implemented using an algorithm named LangId. With the use of Huggingface, NER models are implemented. Named entity matching is deployed using LIG3 which is sourced from the Aydos python package. Eponymous disease detection, or medical term detection, is implemented by querying SNOMED-CT. Which is situated in a separate docker container. And lastly the de-identification is done using a preset list of names which is not found in any medical database.

The treatment is optimised with the implementation of multiprocessing in order to utilise all processing cores of the system. The python framework that was used to quickly create a database and REST api endpoints is Django [4]. This framework was also used in the original Care2Report system. This makes it possible to fully integrate the de-identification pipeline into the Care2Report system on a code level.

4.3 Requirements

To test the quality of the treatment, evaluate it, a number of requirements are defined and tested. These requirements are defined below.

- Multilingual
- Real-time & Lightweight

For each requirement, three pieces of information are given in order to define the requirement. First, the relation to the challenges presented above, together with its relation to the stakeholders and goals. This places the requirement in the context of this research paper. Second, the scope of the requirement is given. Each requirement can only be tested when a clear scope is defined in order to limit the number of variables that needs to be accounted for. After the requirement has been defined, a method is presented which is used to validate the requirement. Thereafter, the acceptance criterion is given.

- Definition
  - Relation to the challenges, stakeholders and goals.
  - Scope of the requirement.
- Validation mechanism
- Acceptance criterion

4.3.1 Multilingual

Definition

The challenge of multilingualism is tackled with this requirement. The goal to be able to de-identify texts with different languages is introduced by the Care2Report R&D team. It asks for a system where different languages can be posted into the system and be de-identified. This is only possible when the NER models, SNOMED-CT database and de-identification placeholders are language specific.

Validation

The prototype system will be presented with input sentences consisting of five different natural languages. The sentence that is fed into the system is a variation of: "Peter Johnson has a sore ankle.". The name used in each language's sentence also is dependent on the language. This results in the test-set depicted in table 4.1.

Language	Name	Sentence
English	Peter	Peter has a sore ankle.
Spanish	Mateo	Mateo tiene un tobillo dolorido.
Dutch	Jan	Jan heeft een pijnlijke enkel.
German	Arend	Arend hat einen wunden Knöchel.
French	Lance	Lance a une cheville douloureuse.

Table 4.1: List of test sentences for multilingualism validation

Acceptance

This requirement is successfully implemented if:

- The correct language is stored for each sentence.
- The correct language model is queried within the Named Entity Recognition task.
- The correct SNOMED-CT database was queried.

The list above describes all the steps where the natural language is of impact on the de-identification process. If all these steps are performed correctly, then the requirement of multilingualism is successfully implemented.

4.3.2 Real-time & lightweight

Definition

Real-time processing and the requirement of the creation of a lightweight pipeline are tested together. This decision has been made as the ability to run the pipeline in real-time is highly dependent on the type of hardware that is used. A faster processor is able to process the sentences more quickly. In order to incentivise the creation of an optimised treatment for this research problem and to make the system practical in its deployment on different systems, the system needs to be able to attain real-time processing on a system that has a low amount of processing speed and RAM memory available.

This requirement is also able to test the treatment for both the developer and end-user goals. The developer wants the treatment to be easily deploy-able and able to run on lightweight systems. The end-user wants the pipeline to be able to run in real-time so that it adds little time to the total processing time of the C2R system.

Validation

This requirement is tested using Dutch sentences. The total test-set consists of four conversations which in turn includes two speakers. Data about the nature of the testdata can be found in table 4.2

ID	Length (syllables)	Name count	Average sentence length
1	213	3	12.22
2	280	4	9.16
3	702	3	15.67
4	852	3	14.66

Table 4.2: Overview of test conversations realtime processing testing.

In order to test whether the pipeline is able to process sentences in real-time, it needs to be fed with data in a frequency which simulates a real-world situation. Research into speech-rate by Dutch speakers concludes that Dutch speakers pronounce around 5,2 syllables per second [16]. The total results ranged between 3.2 (-38.5%) and 6.3 (+21.2%) syllables per second. A large annotated speech database of Dutch language was used to conduct this research. The *Corpus Gesproken Nederlands*. These rates are the base of the required speed that the treatment should be able to handle. In order to test if the system is able to handle higher conversational speeds, the system needs to be tested a higher speaking rates. This research will use an average speaking rate of 6.3 syllables per second.

Acceptance

This requirement is full-filled when: *"The pipeline can process input data with an average speed of 6.3 syllables per second on a Raspberry Pi 4 (model B, 8GB RAM)."*

Chapter 5

Treatment Evaluation

This chapter will look at the implemented treatments and tests their quality. They are tested against the thresholds and requirements which were defined in chapter 4. For each requirement, a short description of the used metric is given. Furthermore, the results for that metric are given and a conclusion explaining whether the requirement was met.

The following requirements were tested:

- Multilingual
- Real-time
- Lightweight

5.1 Multilingual

This requirement is successfully implemented if:

- ✓ The correct language is stored for each sentence.
- ✓ The correct language model is queried within the Named Entity Recognition task.
- ✓ The correct SNOMED-CT database was queried.

5.1.1 method

In order to test whether the treatment is multilingual, the pipeline is presented with sentences from 5 different languages 5.1. If the pipeline is able to detect each language successfully, and the pipeline is able to perform the steps which are natural language dependent successfully, then this requirement is fulfilled. The script that is used to test this requirement can be found in Appendix A.4. To find out whether the pipeline used the correct model and SNOMED-CT release, slight modifications were made to the pipeline in order to show this data in the system output during run-time.

Language	Name	Sentence
English	Peter	Peter has a sore ankle.
Spanish	Mateo	Mateo tiene un tobillo dolorido.
Dutch	Jan	Jan heeft een pijnlijke enkel.
German	Arend	Arend hat einen wunden Knöchel.
French	Lance	Lance a une cheville douloureuse.

Table 5.1: List of test sentences for multilingualism validation

5.1.2 results

Language	Name	Sentence	Detected language	System output correct
English	Peter	Peter has a sore ankle.	English	✓
Spanish	Mateo	Mateo tiene un tobillo dolorido.	Spanish	✓
Dutch	Jan	Jan heeft een pijnlijke enkel.	Dutch	✓
German	Arend	Arend hat einen wunden Knöchel.	German	✓
French	Lance	Lance a une cheville douloureuse.	French	✓

Table 5.2: List of test sentences for multilingualism validation with the detection results.

5.1.3 conclusion

In conclusion, the treatment is able to successfully detect the language. Store it. And use that information to perform the sequential steps in the context of the stored natural language. This requirement has been met.

5.2 Real-time & Lightweight

This requirement is successfully implemented if:

- ✓ The pipeline can process input data with an average speed of 6.3 syllables per second.
- ✓ The pipeline can perform on a Raspberry Pi Model 4 - 8GB RAM

5.2.1 method

The pipeline will be presented with a stream of text. This stream will consist of four different "conversations". These conversations are created using text-to-speech software which is embedded in Microsoft Word. Grammatical and spelling errors were recovered manually in order to ensure perfect transcription text as input data. The pipeline is fed with sentences of different lengths where the average amount of syllables equals 6.3 per second throughout the entire "session". To ensure correctness, a testing script was designed (Appendix A.2).

The steps that were undertaken for each run in order to conduct real-time processing capabilities measurements are the following:

Step 1. A Raspberry Pi was setup with the required software. This consists of the following:

- MySQL
- django-admin
- nginx

Step 2. The GitHub repository was retrieved which contains the treatment. Required python packages were installed. A list of these packages can be found in appendix A.1.

Step 3. The API endpoint of the treatment was launched. Which made the treatment ready to receive to-be de-identified sentences.

Step 4. The test code was run (appendix A.2). Which simulates a real-world scenario where the sentences contained in the test-texts (appendix A.5) were sent to the pipeline where they were de-identified.

Step 5. During the test-run, timestamps of the processing start and end times were collected and added to the database entry of each sentence. This information was retrieved after completion of the test code (appendix A.2).

The pipeline itself needs to query SNOMED-CT during runtime in order to perform the Eponymous disease queries of detected entities. This was omitted in the testing setup as it was not possible to deploy the SNOMED-CT database due to technical limitations and licensing issues.

5.2.2 Results

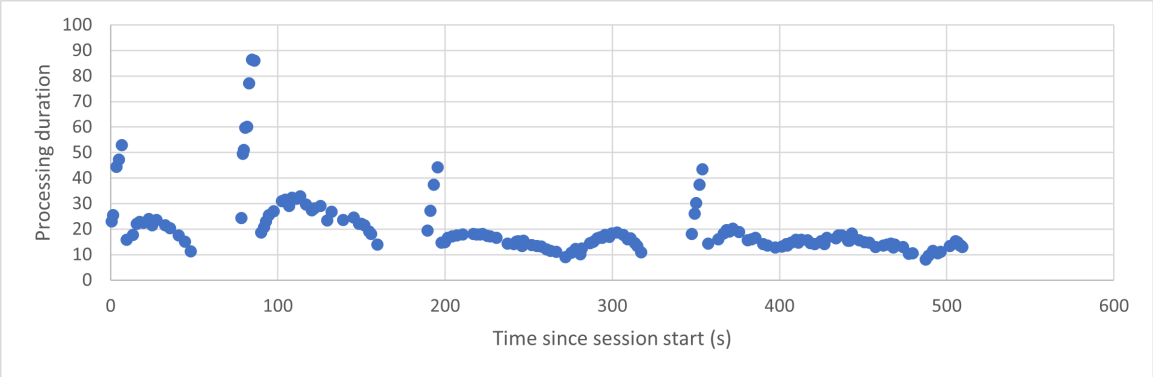

Figure 5.1: Processing speed of sentences

This figure shows the processing time needed for each sentence individually. The y-axis displays the processing duration. The x-axis shows the seconds between the session start and the receival of this particular sentence. Each dot represents a sentence that was processed.

There are four groups of sentences. Each group represents one of the simulation texts. For each simulation text, it can be observed that the first five sentences take longer to be processed. Thereafter, the processing speed dramatically increases. The overall average processing speed is 20 seconds per sentence for all four sessions. The results for each different session are presented below.

Conversation	AVG	AVG IGN	Total
1	24.8 sec	19.9 sec	47.2 seconds
2	34.2 sec	25.3 sec	81.4 seconds
3	16.5 sec	15.1 sec	127.5 seconds
4	16.0 sec	14.6 sec	161.8 seconds

Table 5.3: Average processing times

In the data above, the average processing time per sentence (AVG) is included as well as a processing time per sentence when the initial 4 to 6 sentences were ignored (AVG IGN). This average is included in the results section as these few initial sentences are outliers. Each session starts off with slower processing of sentences after which the processing speed normalises towards the average of 20 seconds per sentence.

A hypothesis for these slow processing speeds at the start of each session, could be that background system tasks that are running are prioritised over the pipeline at first. But after spawning each de-identification process, the system prioritises the pipeline over background processes thus becoming faster as tasks are picked up faster.

During the design phase, attempts were made to remove these "outlier" sentences by priming the pipeline with fake sentences. The pipeline was presented with a sentence every 2 seconds which would keep the pipeline occupied thus removing the initial slowdown. This did not work.

Within each of the four conversations that were processed, a pattern can be observed. The test algorithm (Appendix A.2) was run five times in order to see if the processing speed pattern that emerged on the first run continued to show up. This was indeed the case. The results are presented in the figure below.

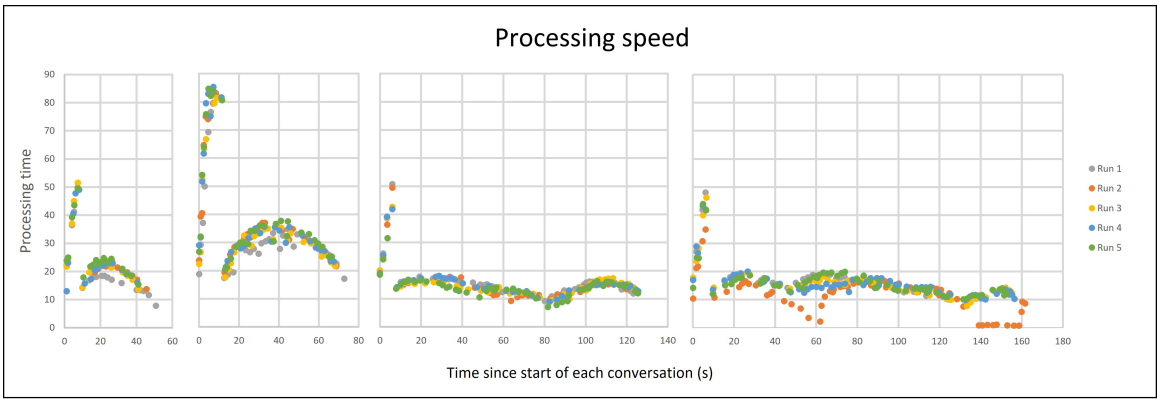

Figure 5.2: Processing speed of sentences - reiterations for pattern observations

In order to find out why this pattern was emerging, the de-identification pipeline was fed several "conversations", where the conversation consisted of a single sentences that was repeatedly fed into the pipeline while still maintaining the average syllable rate of 6.3 syllables per second. The length of the sentences was differentiated throughout the different conversations. There was always one entity present in the sentences. The code used to test why this pattern occurs can be found in Appendix A.3

- Sentence count per conversation: 20
- Syllable rate sent to pipeline: 6.3
- Sentence content fixed within each conversation.
- Lengths of test sentence were a multiple of 6.3.

Conv.	syl. #	sentence	AVG
1	1	Tom.	81.1 sec
2	6	Hallo mijn naam is Tom.	38.4 sec
3	13	Hallo mijn naam is Tom en ik roei iedere dag.	18.7 sec
4	25	Hoi mijn naam is Tom en ik roei iedere dag op mijn favoriete roeivereniging Triton.	6.5 sec

Table 5.4: Pattern discovery test set

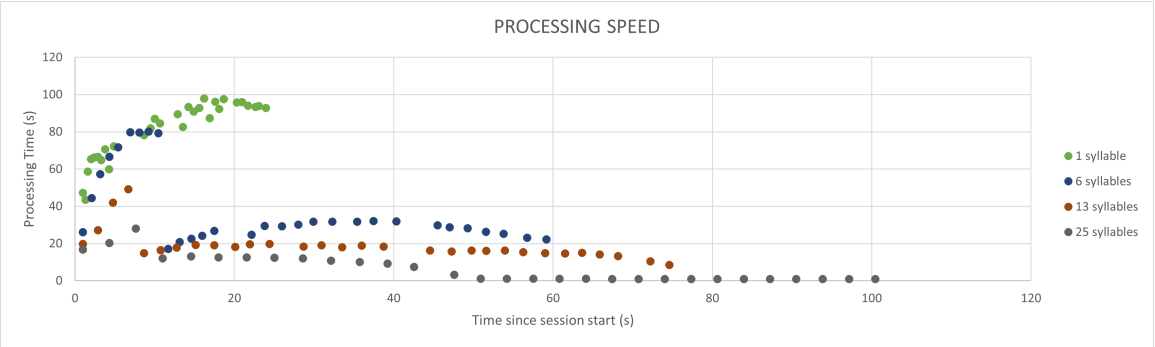

Figure 5.3: Processing speed of sentences - pattern exploration

The average processing time needed for each sentence in much higher when the sentences are shorter. This means a higher sentence length (consisting of more syllables) is of positive influence on the processing speed. This also clarifies the processing patterns that emerge. The first two conversations consist of relatively short conversations, syllable count wise. When the metadata of the conversations and the average processing time per sentence is aligned in the table below. This relation can be observed, also when the first initial few sentences are ignored in the average processing duration.

ID	syllable count	Average sentence length	AVG	AVG IGN
1	213	12.22	24.8	20.4
2	280	9.16	34.2	25.8
3	702	15.67	16.5	15.2
4	852	14.66	16.0	14.6

Table 5.5: Comparison sentence length and Average processing time

5.2.3 Conclusion

Looking at the results, it can be concluded that the treatment is able to process the sentences in real-time. In each of the four sessions the processing finishes without the need of a backlog. This means this requirement is successfully attained. The processing speed is dependent on the length of the sentences. The longer the sentences contained in the conversation. The more efficient the system is able to process the entire conversation.

✓ The pipeline can process input data with an average speed of 6.3 syllables per second.

✓ The pipeline can perform on a Raspberry Pi 4 Model B - 8GB RAM

Chapter 6

Discussion

The result of this research is a de-idenfitication pipeline which is able to process multilingual sentences on a lightweight system in real-time. However, there are still a number of things that should be considered in follow up research in order to create a perfect de-identification pipeline.

SNOMED deployment - During the testing of the developed pipeline, it appeared impossible to deploy SNOMED. The system had to be deployed in a docker architecture which was unsuccessfully, even after inquiring for support. Therefore the SNOMED implementation was omitted in the test-setup. Future research concerning the de-identification pipeline should aim to have this database implemented in order to get a more qualitative de-identification output.

Other personal identifiers - As was discussed in section 3.3, this pipeline only replaces names in the care2report system input. However, other identifiers also need to be replaced in order to create a truly anonymous transcription. Future research should look into the possibilities of incorporating an algorithm which is able to replace identifiers using their context. This might allow a future de-identification pipeline to attain true anonimisation which would allow far broader use of the transcripts in other systems or to train C2R components without the concern of disclosing protected information.

NER model accuracy - In this research, the quality of the NER model was ignored. This was done as this research aimed to solely discover the possibility of creating a de-identification pipeline. However, future research should consider this factor as the degree of de-identification is highly dependent on it. If the model fails to detect a name, they are simply ignored which not only harms the privacy of the end-user but also impairs the care2report processing as a single real-world entity is denoted using two completely different names.

Huggingface implementation - The model repository Huggingface is used with the purpose of implementing different NER models. The implementation itself is very promising as the models are deployed on the server immediately upon launch. However, a high degree of coupling is needed in order to have the pipeline word with different models. This results in the need to change back-end programming depending on the model that is used. This defeats the entire purpose of the Huggingface implementation as it was done to dynamically deploy different models. Therefore, future research should also look into a mechanism which would allow truly hotswappable model use.

Further optimisation - With the findings of section 5.2.2, the pipeline can be further optimised by combining different sentences during run-time in order to optimise model throughput. This would possibly even allow to reduce processing times to 2 seconds per sentence. Initial tries proved successful. However, after full implementation the system failed to respond anymore and no solution was found. Though this seems to be a very simple way of greatly improving the system.

Another research angle to further optimise the treatment is by eliminating the initially slow sentence processing. The attempt that was made to tackle this issue, was to feed the pipeline with so-called primers. These would keep the system activated and therefor remove the initial slowdown of the system. However, this did not work. Further research into the cause of this slow initial processing also did not prove to be fruitful. Therefore, this challenge can be researched further by other researchers.

Chapter 7

Conclusion

In conclusion, a pipeline, which treats the technical research problem, has been created.

> *"**How to** create a de-identification pipeline **that** works with different languages, in real-time and is lightweight **so that** the names of end-users are replaced with real-world identifiers to improve their privacy **in** the Care2Report pipeline?"*

There are still a lot of problems which need to be tackled in order to create a perfect de-identification pipeline of which the output is perfect and therefor anonimized. But, it can be concluded that, with the use of existing technologies, a pipeline that can de-identify medical transcriptions can be created.

> *MRQ*: **"How can a pipeline be constructed to de-identify medical transcriptions, which can be used with the Care2Report system?"**

The pipeline was constructed by applying the Wieringa design cycle, a scientific process of solution development for a design problem found in the information technology domain. This design process resulted in a treatment of the technical research problem that was uncovered within the design cycle. The pipeline can be constructed using several already existent natural language processing algorithms and models.

By making the pipeline available through locally deployed API endpoints, the solution can be implemented within all sorts of system architectures. Also inside the Care2Report architecture. By copying the base web-server framework used for HTTP interactions, the treatment is also easily maintainable by the current development team of Care2Report.

RQ1: "What steps need to be performed in order to construct a pipeline to automatically de-identify transcriptions?"

The steps that need to be performed in order to automatically de-identify transcriptions are the following:

1. Natural Language Detection (NLD), the detection of the natural language of the transcript.
2. Named Entity Recognition (NER), the detection of entities within the transcript.
3. Named Entity Matching (NEM), the connection of differently denoted but same real-world entities.
4. Eponymous disease detection, the detection of named entities named after a disease.
5. De-identification, the replacement of named entities in each sentence.

RQ2: "How can entities be detected in perfect transcription data?"

With the use of NER models, real-world entities can be detected within perfect transcription data. The implementation of a so-called model repository within the code also allows the treatment to detect entities in a wide variety of natural languages.

RQ3: "How can the de-identification process be performed in real-time?"

By implementing multiprocessing technology, multiple cores of the server can be utilised. This optimises the script greatly. All NLP models that are used within the treatment are also pre-loaded in order to further optimise the treatment. These two optimisations make it possible to run the de-identification process in real-time on a Raspberry Pi.

RQ4: "Can the de-identification pipeline perform in real-world simulations?"

The pipeline was able to perform its de-identification task in a real-world simulation. This simulation was created by running a test script which simulates a situation in which the de-identification pipeline would receive four different conversations.

Bibliography

[1] Accuracy, Recall, Precision, and F1 Score, Jul 2021. [Online; accessed 18. Nov. 2021].

[2] multiprocessing — Process-based parallelism — Python 3.10.0 documentation, Nov 2021. [Online; accessed 19. Nov. 2021].

[3] abydos, May 2022. [Online; accessed 22. May 2022].

[4] The web framework for perfectionists with deadlines | Django, May 2022. [Online; accessed 22. May 2022].

[5] L Karl Branting. A comparative evaluation of name-matching algorithms. In *Proceedings of the 9th international conference on Artificial intelligence and law*, pages 224–232, 2003.

[6] Ido Cohn, Itay Laish, Genady Beryozkin, Gang Li, Izhak Shafran, Idan Szpektor, Tzvika Hartman, Avinatan Hassidim, and Yossi Matias. Audio De-identification: A New Entity Recognition Task. *arXiv*, 3 2019.

[7] Velzen J Draief M, Bheemaiah K. What's next for data, blockchain and quantum computing? | Capgemini Invent, Jan 2022. [Online; accessed 2. Feb. 2022].

[8] John S Garofolo, Cedric GP Auzanne, Ellen M Voorhees, et al. The trec spoken document retrieval track: A success story. *NIST SPECIAL PUBLICATION SP*, 500(246):107–130, 2000.

[9] Mohamed Hatmi, Christine Jacquin, Emmanuel Morin, and Sylvain Meigner. Incorporating named entity recognition into the speech transcription process. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (Interspeech'13)*, pages 3732–3736, 2013.

[10] Mohamed Ameur Ben Jannet, Olivier Galibert, Martine Adda-Decker, and Sophie Rosset. How to evaluate asr output for named entity recognition? In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[11] Francis Kubala, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. Named entity extraction from speech. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 287–292. Citeseer, 1998.

[12] Marco Lui and Timothy Baldwin. Cross-domain feature selection for language identification. In *Proceedings of 5th international joint conference on natural language processing*, pages 553–561, 2011.

[13] Marco Lui and Timothy Baldwin. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30, 2012.

[14] Lientje Maas, Mathan Geurtsen, Florian Nouwt, Stefan Schouten, Robin Van De Water, Sandra Van Dulmen, Fabiano Dalpiaz, Kees Van Deemter, and Sjaak Brinkkemper. The care2report system: automated medical reporting as an integrated solution to reduce administrative burden in healthcare. 2020.

[15] mzsanford. cld, Nov 2021. [Online; accessed 16. Nov. 2021].

[16] Hugo Quené. Andante of allegro. *Verschillen in spreektempo tussen Vlamingen en Nederlanders. Onze Taal*, 77:179–181, 2008.

[17] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[18] P. Richter-Pechanski, S. Riezler, and C. Dieterich. De-Identification of German Medical Admission Notes. *Stud. Health Technol. Inform.*, 253:165–169, Jan 2018.

[19] Priya Shree. GPT models explained. Open AI's GPT-1,GPT-2,GPT-3 | Walmart Global Tech Blog. *Medium*, Dec 2021.

[20] shuyo. language-detection, Nov 2021. [Online; accessed 16. Nov. 2021].

[21] Chakkrit Snae. A comparison and analysis of name matching algorithms. *International Journal of Applied Science. Engineering and Technology*, 4(1):252–257, 2007.

[22] Chakkrit Snae and BM Diaz. An interface for mining genealogical nominal data using the concept of linkage and a hybrid name matching algorithm. In *Journal of 3D-Forum Society*, volume 16, pages 142–147, 2002.

[23] Katsuhito Sudoh, Hajime Tsukada, and Hideki Isozaki. Incorporating speech recognition confidence into discriminative named entity recognition of speech data. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 617–624. Unknown publishers, Jul 2006.

[24] Amogh Kamat Tarcar, Aashis Tiwari, Vineet Naique Dhaimodker, Penjo Rebelo, Rahul Desai, and Dattaraj Rao. Healthcare ner models using language model pretraining. *arXiv preprint arXiv:1910.11241*, 2019.

[25] Trey314159. TextCat, Nov 2021. [Online; accessed 16. Nov. 2021].

[26] Özlem Uzuner, Yuan Luo, and Peter Szolovits. Evaluating the State-of-the-Art in Automatic De-identification. *J. Am. Med. Inform. Assoc.*, 14(5):550–563, Sep 2007.

[27] R. Wieringa. *Design Science Methodology for Information Systems and Software Engineering.* Springer, 2014.

[28] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[29] Hemant Yadav, Sreyan Ghosh, Yi Yu, and Rajiv Ratn Shah. End-to-end named entity recognition from english speech. *arXiv preprint arXiv:2005.11184*, 2020.

Appendices

Appendix A

Code

A.1 Python Requirements

- packaging
- flair
- torch
- langid
- wget
- django
- fuzzywuzzy
- abydos
- mysqlclient
- mysql-connector-python
- git+https://github.com/huggingface/transformers.git

A.2 Realtime test code

```
# This list contains all testing conversations.
convos = [conv1, conv2, conv3, conv4]

# This list contains four sessions which are used for each conversation.
seshs = [sesh1, sesh2, sesh3, sesh4]

# we list over de four conversations.
# Testing one after another with 30 seconds in between the three conversations.
for conv in range(0, 4):

    # take the conversation from the list
    convo = convos[conv]

    # loop over the sentences in the conversation.
    for sentence in convo:

        # remove any leading spaces and spawn a sentence submission process.
        # this is used to ensure the syllable rate of 6.3
        asyncio.run(sender(sentence.strip(), seshs[conv]))

        # count the syllables in each sentence
        syllables = syllable_count(sentence)
        # wait until sending the next sentence in order to maintain a rate of 6.3
        time.sleep(syllables / 6.3)

    # wait 30 seconds in between each conversation
    time.sleep(30)
```

A.3 Syllable pattern test code

```
# store every test sentence in a list.
conv = [
    "Tom",
    "Hallo mijn naam is Tom",
    "Hallo mijn naam is Tom en ik roei iedere dag",
    "Hallo mijn naam is Tom en ik roei iedere dag "
    "op mijn favoriete roeivereniging Triton"
]

# loop over the four different sentences.
for i in range(0, len(conv)):

    # create a new session in the pipeline for each sentence as
    # these are considered a conversation.
    sesh = requests.post(
        "http://192.168.1.175:8001/sessiondata?type=open",
        json={
            "sessionID": str(datetime.datetime.now().timestamp() + 4)
        }).json()["response_data"]["ID"]

    # loop over the sentence 30 times.
    for j in range(0, 30):

            # take the sentence from the list
            sentence = conv[i]

            # remove any leading spaces and spawn a sentence submission process.
            # this is used to ensure the syllable rate of 6.3
            asyncio.run(sender(sentence.strip(), sesh))

            # count the syllables in each sentence
            syllables = syllable_count(sentence)
            # wait until sending the next sentence in order to maintain a rate of 6.3
            time.sleep(syllables / 6.3)

    # wait two minutes in between the different conversations.
    time.sleep(120)
```

A.4 Multilingualism test code

```
# store the different sentences is variables.
sentence_EN = "Peter has a sore ankle."
sentence_SP = "Mateo tiene un tobillo dolorido"
sentence_NL = "Jan heeft een pijnlijke enkel"
sentence_DE = "Arend hat eine wunden Knockel"
sentence_FR = "Lance a une cheville douloureuse"

# make them into a list.
sentences = [sentence_DE, sentence_FR, sentence_SP, sentence_NL, sentence_EN]

# loop over the sentences
for sentence in sentences:

    # create a new session for eacht sentence
    sesh = requests.post(
        "http://192.168.1.175:8001/sessiondata?type=open",
        json={
            "sessionID": str(datetime.datetime.now().timestamp() + 4)
        }).json()["response_data"]["ID"]

    # remove any leading spaces and spawn a sentence submission process.
    # this is used to ensure the syllable rate of 6.3
    asyncio.run(sender(sentence.strip(), sesh))
```

A.5 realtime test text

A.5.1 text 1

Sentence count	18
Syllable count	213
Language	Dutch
Word count	168
Named Entities	3

Table A.1: Details about the testset.

Hoi Ik ben Jacco. Hoi aangenaam, Ik ben dr Vreeswijk, Wat kan ik voor je doen. Ik heb last van mijn scheenbeen. Ik heb 1 grote rode plek en ik weet niet echt wat het is. Oké, Laten we dan eens gaan kijken waar je last van hebt, meneer Broeren. Als je even naar de behandeltafel wandelt en daar gaat zitten, kijken ik even. Oké, Dit is inderdaad een rode plek. Doet het pijn als ik hier duw. Klein beetje het zeurt wat. En Als ik hier duw. Ja dat doet pijn. Oké ga er maar weer zitten. U heeft een onderhuids ontsteking. Mocht die groter worden, of als u last van koorts krijgt, dan moet u even bellen en dan moeten we even opnieuw kijken. Misschien moet u dan antibiotica. Maar vooralsnog nog ziet het er goed uit. Omcirkel de plek die rood is en doe dat dan over 24 uur nog een keer. Als het dan dus groter is geworden, dan mag u even contact opnemen.

A.5.2 text 2

Sentence count	29
Syllable count	280
Language	Dutch
Word count	215
Named Entities	4

Table A.2: Details about the testset.

Goedemorgen. Goedemorgen. Ik ben Jan, hoi. Hoi, Ik ben dokter De boer. Wat kan ik voor u doen. Ik heb last van mijn oor. Oké, dan gaan we even kijken. gaat u maar op de behandeltafel zitten, dan pak ik even mijn oor kijkapparaat erbij. Nou oké, Het is rood van binnen. Heeft u koorts. Nee, Ik heb geen koorts alleen oorpijn. En is er vocht uit de oor komen lopen Meneer De Vries. Nee, dat heb ik nog niet gemerkt. Vorige keer was het wel zo. Nou en wanneer was de vorige keer. Hmmm. Ongeveer een week geleden. Oh, Dat is heel snel op deze keer. Hoeveel pijn deed hij toen. Nou, Ik heb toen een paracetamolletje genomen om in slaap te vallen. Oké. En heeft u nu iets geslikt. Nee maar ik zat daar wel over na te denken. O. Nou, u heeft gewoon een ontsteking op het moment, maar Omdat u het een week hiervoor ook al had, lijkt het niet zo goed weg te gaan. Dus ga ik u op een antibiotica kuur zetten. Oké, ik maak even een verwijsbriefje voor u. Uw naam was. Jan De Vries. Oké, krijgt u een verwijsbrief mee namens dr. Piet de boer. Oké, dank u wel, dan ga ik nu even langs de apotheek.

A.5.3 text 3

Sentence count	45
Syllable count	702
Language	Dutch
Word count	575
Named Entities	3

Table A.3: Details about the testset.

Het was echt super chill. Het was mooi weer, heel veel koffie drinken, was echt chill. En ja dus ook heel veel getraind, want we waren op trainingskamp. 17 keer in 10 dagen hebben we volgens mij getraind. Heel veel, was ook wel goed te doen aan het eind. Nog veel tapas gegeten. Ja, We hebben heel veel tapas gegeten. Maar met dat vele trainingen kreeg je geen last van je rug Jacco. Op de 4/5 dag wel last van mijn rug, maar dat trok ook wel weer weg. Toen hebben we volgens mij 20 km training iets korter moeten varen, maar ja, Dat was op zich ook niet zo'n probleem, want we trainden sowieso al super veel. En, Het was echt super chill. Dat Malte op een motorboot zat, hij coachte echt. Dat was heel Nice. En, je hebt daar dus coach boten die varen dan achter je boot aan. En, daar staat dan je coach op. Dus die kan dan eigenlijk heel erg goed om je heen varen om te kijken wat je fout doet. Dat helpt enorm met coaching en ze kunnen ook heel Nice filmen, dus je hebt gewoon de hele tijd filmpjes van de zij voor en achterkant waar je heel goed kan zien wat je zelf ook fout doet. En Dat is een stuk makkelijker om dingen te verbeteren die je coach dan benoemd. Omdat je het in één keer zelf ziet. Hoe verbleven jullie daar. We zaten in het huisje daar met 9 of 8. Volgens mij. Er was nog wat gezeik, omdat er twee perse met elkaar in een kamer wilden, zodat ze niet met andere Mensen In de Kamer hoefden. Maar uiteindelijk was het wel opgelost en was nog heel gezellig. Was wel een kleine keuken, dus koken was vrij ingewikkeld. En de ruimte was sowieso zelf niet heel erg groot, dus ja, Het was wat krap soms. Maar op zich was dat ook allemaal geen ramp. En, Het was wel heel gezellig en We hadden ook allemaal fietsen. We hadden allemaal fietsen die we konden gebruiken om van het huisje naar de baan te gaan, want Dat is toch wel een goed halfuur lopen Als je het te voet moest doen. En die fietsen, die kocht je dan eigenlijk over van een lokale fietsenmaker daar voor € 100 ofzo. En dan bracht je die aan het eind van de dag terug aan het eind van de week. En dan betalen ze je gewoon € 70, terug, of de helft in ieder geval. En dat. Nou ja, dat bedrag krijg je dan gewoon terug en dan betaal je dus effectief veel minder dan wanneer je een fiets huurt voor de hele week. Alleen op de eerste twee dagen hadden we nog geen fiets. In sevilla hebben ze dus allemaal stepjes. Een step die je kan gebruiken. En dan betaal je na gebruik. Alleen € 1 per minuut ofzo. Dus Dat is best wel veel, maar Als je niet heel lang onderweg bent. Oh nee, 20 cent per minuut. We hadden trouwens al onze spullen in de boot gestopt. Daardoor had je niet dat je ook nog een ruimbagage tas mee moest nemen, omdat je gewoon je kleding In de boot stopt. En, die kan je dan daarna daar ook weer er uit halen. Dus Dat is. Ja, je hoeft niet kleren niet allemaal In het vliegtuig mee te nemen. Wat ook chill is. Dus Dat was de trip naar Sevilla.

A.5.4 text 4

Sentence count	60
Syllable count	852
Language	Dutch
Word count	638
Named Entities	3

Table A.4: Details about the testset.

Goedemiddag, Hallo dokter Horstman. Waarom ben je hier. Nou, Ik heb een klein ongelukje gehad met een pan met pasta. OK vertel, wat is er gebeurd. Ik was op zaterdagmiddag pasta aan het koken voor na een roeiwedstrijd. En Ja, toen liep ik met een pan met heet water van het fornuis naar de grote steen om pasta af te gieten en toen brak de linker pannen hendel af. En ja toen kreeg ik al het hele kokende water over mijn bovenbeen heen. Dus Dat is vrij jammer. Oké, en wat heb je toen gedaan. Heb je toen direct gekoeld? Ja, Ik heb toen eerst een half uur koud lauw water over de wond heen Laten lopen. En toen ben ik daarna naar de huisartsenpost gegaan bij het diak en daar hebben ze toen naar de wond gekeken. Het verbonden. En nou ja. Toen is er paraffine gaas overheen gelegd en nog wat niet klevend, ander gaas wat niet ingevet was. En daar bovenop dan gewoon verband. Maar ja, Het was paasweekend, dus ik kon ook niet langsgaan bij de apotheek om dat verband te verwisselen, dus. Nee, Daarom moest ik ook hierlangs nog en van de huisarts post moest ik sowieso nog langs voor een controle. Omdat ze gewoon wilden dat er nu nog naar gekeken werd. Dus ja. Oké, dan ga ik even kijken naar je been. Ik ga eventjes het verband eraf halen. Is het heel gevoelig. Ja, oké. Nou, Als ik er zo naar kijk, dan ziet het er goed uit. Ik zie geen ontstekingen. Het ruikt ook niet ontstoken, dus Dat is goed. Ik wil nog even kijken of je wel goed gevoel hebt In de hele wond. Dus ik pak even een wattenstaafje om te kijken. Met wie ging je roeien dit weekend. Ik roeide met Wouter in een dubbel op de voorjaars regatta, maar ja, daar kon ik niet meer op starten op zondag met deze wond, helaas. Dat was een beetje jammer. De race ging wel heel goed op zaterdag. We zijn er nu algemeen derde in het klassement, dus Dat is wel nice. Voel je dit. Ja. En dit. Ja ook goed, oké, dan is het gevoel in ieder geval goed, dan gaan we er nu zalf op smeren. Deze zalf die doodt bacteriën om ontstekingen te voorkomen. En daaroverheen, doen we dan hetzelfde. Paraffine gaas zoals dat ze bij de huisartsenpost hebben gedaan en nog wat ander gaas en dan. Krijg je waarschijnlijk een elastisch verband daar weer overheen, wat zorgt dat het verband niet af gaat zakken. Met al dat vet ertussen gebeurt het nogal snel. Weet je toevallig welke huisarts je had gezien In het diak. Ja, huisarts van der Mark. Oké. Dan ga ik even de update van hem in jouw medisch dossier bekijken wat hij er ook van vond. Maar ik vind het dus goed uitzien, mag je over twee dagen terugkomen voor controle. En dan kijken we even hoe je wond zich ontwikkelt. Ik wil vooral heel goed In de gaten houden dat het niet gaat ontsteken, dus op het moment dat je koorts krijgt of een een geelgroene afzetting op de wond ziet, dan mag je onmiddellijk naar de huisartsenpost bellen. Want dan moeten we daar gelijk naar kijken. Maar voor nu ziet het er heel goed uit, dus ik wens je vooral veel sterkte. Neem ook goede pijnstilling, je mag gewoon maximale hoeveelheid slikken, dus Dat is. 400 mg Ibuprofen en 1000 mg paracetamol. En dat kan je gewoon net zo lang slikken totdat de pijn afneemt, maar je bent gezond, dus ik vermoed dat het herstel relatief snel gaat. Ik denk dat je over ongeveer 1,5 week wel weer zou moeten kunnen lopen en roeien. Oké, dank u wel. Tot ziens en tot morgen dan. Ja. Tot volgende keer, joe. Beterschap.

A.6 Replacement names

1. Bobby
2. Noa
3. Daan
4. Alex
5. Jessie
6. Jip
7. Anne
8. Ollie
9. Sam
10. Guus