Automated Extraction of Time References From Clinical Notes in a Heart Failure Telehealth Network

Fabian Wiesmüller[1,2], Alphons Eggerth[1], Karl Kreiner[1], Dieter Hayn[1], Sten Hanke[2], Bernhard Pfeifer[1], Gerhard Pölzl[3], Tim Egelseer-Bründl[4], Günter Schreier[1]

[1]AIT Austrian Institute of Technology GmbH, Graz, Austria
[2]FH JOANNEUM GmbH, Graz, Austria
[3]Department of Internal Medicine III, Cardiology and Angiology, Medical University Innsbruck, Innsbruck, Austria
[4]Landesinstitut für Integrierte Versorgung – LIV Tirol, Innsbruck, Austria

Abstract

Heart failure (HF) is one of the biggest concerns for health care systems in developed countries. To support the long-term treatment of HF patients, the Austrian Institute of Technology implemented a HF telehealth network called "HerzMobil". While most data within this network are stored in a structured format, health care professionals can also communicate via clinical notes in free text format. These notes are hardly ever analyzed automatically, even though a large number contains valuable information for the patient's treatment process. With currently more than 20,000 notes stored in the system, an automatic approach is beneficial to spare manual screening time. One important step in this process concerns the extraction of time references from the notes. This information could, for example, be used to match the time references with events from the same note. Therefore, two Python scripts were developed to: extract time references from the notes (Script A) and subsequently calculate the corresponding dates (Script B). Script A was compared to an already existing Python library and achieved superior results for all calculated key figures. The time calculation algorithm of Script B achieved an accuracy of 75.34%. These scripts could be implemented in the HerzMobil network to provide additional information for the treatment process and further improve the telehealth system.

1. Introduction

1.1. Background

With an increasingly old population, chronic diseases are on the rise. With an estimated prevalence of one to two percent in adults, heart failure (HF) is one of the biggest concerns for health care systems in developed countries [1]. The 12-month all-cause hospitalization rates are 44% for hospitalized HF patients and 32% for ambulatory patients. These rates are amongst the highest for the elderly population [1,2].

To improve the treatment management for HF patients and to reduce the hospitalization rates, a HF telehealth network called "HerzMobil Tirol" was established in the Austrian province of Tyrol [2]. Patients within HerzMobil transmit data like e.g. daily measurements of their vital parameters, like blood pressure or heart rate, in a structured form. Amongst other communication methods, healthcare professionals exchange information about a patient's status of health via clinical notes in the form of free text [2].

The majority of the notes contains information directly related to a patient's treatment process, such as a change in medication or an adaptation of a threshold value. Thus, analysing these notes can provide valuable information for the involved health professionals.

At the moment, more than 20.000 notes written in German language were been created in the HerzMobil network. Due to the large volume of notes, an automated analysis approach is beneficial to reduce manual screening. Additionally, once an algorithm for an automated analysis is developed, it can be slightly adapted and deployed in various other telehealth solutions provided by the AIT.

1.2 State-of-the-art

Efforts towards the analysis of clinical notes have already been made, like a system developed by Hebal et al. [3]. This system, however, relied on clinical notes in a structured form and complete free text could not be analyzed. Other already existing algorithms for extracting information from clinical notes used domain ontologies to recognize and detect named entities [4]. Zhang et al. focussed on the use of Statistical Language Modeling, where a probability is assigned to a specific group of words

within the examined set of notes [5]. A paper by Lee et al. implemented a system based on a Support Vector Machine to extract time information and corresponding events from clinical notes [6]. A recent study by Gonzalez-Hernandez et al. used natural language processing (NLP) techniques such as query formulation, keyword selection or Latent Dirichlet Allocation to mine health data from social media texts [7]. Eggerth et al. used NLP to develop classifiers for automated categorization of clinical notes [8]. However, hardly any research has been done on a date extraction algorithm in clinical notes of HF telehealth patients.

1.3 Objectives

This paper makes first efforts towards an automated analysis of the HerzMobil notes by developing a time reference extraction algorithm and subsequently calculating actual dates for the corresponding time references. These two algorithms could be implemented in the HerzMobil system, to give for example real time feedback to healthcare professionals while entering a note. This feedback could consist for example of a suggested calendar entry or an automated allocation of tasks.

2. Methods

2.1. Dataset

All developments were made within an experimental setup. In this setup, the raw notes have been de-identified and split into sentences by AIT scientists, using certain delimiters. The resulting dataset contained 8,832 individual text snippets which originated from 3,952 raw notes, concerning 105 HF patients.

In the following, the word "note" refers to such a text snippet on the sentence layer.

2.2. General structure

To achieve an automated time reference extraction algorithm, two different Python scripts were created. The purpose of these scripts was to extract time references from the notes (Script A) and subsequently calculate their corresponding dates (Script B). Since the amount of available data was not sufficient for machine learning approaches, the scripts used rule based regular expressions to filter time references. These regular expressions were used to match a so-called pattern with a character or a string [9].

2.3. Script A – Time reference extraction

In Script A, 18 different regular expressions were used to match the correct words and right number of digits, dots, whitespaces, etc. with the contents from the notes.

Whilst some expressions for unsophisticated and common terms like *heute* (*today*) or *gestern* (*yesterday*) were created once at the start of the script, others had to be dynamically generated at runtime. This was the case for e.g. months and weekdays. In addition to these phrases, the script filtered specific times of day and added them to the filtered expression. This extension was appended to every regular expression, where it seemed necessary. (Terms like e.g. “one year ago” were excluded from this procedure).

To give an example, the regular expression, which covered the simplest expression, namely *heute* (*today*), was the following: *(?:^|\s?)heute(?:\s?\w*\s\d{0,2}(?:\.|\:|\s)?\d{1,2}\s?(?:h|uhr))?(?:\s|$|!)*. Within the first and last parentheses were terms for the start and end of the expression, like optional whitespaces or line breaks. The literal part of this pattern aimed at the actual time reference *heute* (*today*). The remaining part of this expression was responsible for filtering the time of day in various variations.

Script A returned a JSON file, containing all notes with at least one time reference. Each note had a newly introduced tag called *timetoken*, where the extracted time reference was stored to. Using this individual tag for the date information made it easy to utilize this information for post-processing steps in Script B. Additionally, saving and displaying the filtered term made it possible to investigate false-positive results.

Script A was compared to an already existing Python library called *parsedatetime*, which seemed to be the most promising date extraction library available. However, this library could not be used in the first place due to multiple reasons. For example, subtle changes like from *letzter monat* (*last month*) to *letzten monat* (*last month*) were enough to cause the expression not to be recognized as a time reference anymore. Additionally, expressions like *letzter monat* (*last month*) always resulted in the 1st of the previous month, regardless of any further specifications of the date.

2.3.1. Evaluation of Script A

For the calculation of the following key figures, a subset of 250 randomly chosen notes with time references and 250 notes without time references were created to ensure a fair comparison. These two subsets were used as input data for both, Script A and the *parsedatetime* library to detect those notes containing time references. Evaluation was done by calculating precision, recall and accuracy for both algorithms.

2.4. Script B – Time calculation

Script A only extracted time references but did not process them any further. Therefore, Script B was

necessary to transform this extracted information into actual times. To do so, it took the entire JSON file from the first step as an input. From each note, only the timetoken with the found expression and the note's timestamp were needed for this processing steps. Assuming that the timestamp was the actual time at which a health care professional added the note to the system, it was possible to use this point in time to add or subtract time, to determine the date mentioned in the note's text. The calculation process was simple for some explicit time references like e.g. *gestern* (*yesterday*), since this would result in the note's timestamp minus one day. Other references, which were more inexplicit, like e.g. *am Wochenende* (*at the weekend*) were more complex, since it had to be determined, whether the corresponding date took place in the future or past. Therefore, certain words were filtered in Script A in addition to the time references, which were used in Script B as indicators for the setting of the date.

2.4.1. Times of the day

A value for the time of day got attached to every calculated date. To obtain a notation capable of intervals, a parameter, which represented half the length of the time interval, was added to the notation format. For every time reference, which included a specific time of day, the time parameter was set to this exact value and the size of the interval was set to zero. For time references referring to an interval, the time parameter was set to the centre of the interval and half of the interval duration was added/subtracted to define the interval.

2.4.2. Evaluation of Script B

For the determination of the accuracy of the calculation algorithm of Script B, a subset of manually calculated dates and intervals from 296 individual time references was compared to the results of Script B. The deviation between the automatically calculated and annotated dates was evaluated via the following formula: $\frac{|Ta-Tr|}{Cr}$. Ta stands for the automatically calculated date, Tr represents the annotated reference value and Cr is half the length of the dates manually annotated interval. The third parameter is used to standardize the results.

The calculated result was considered correct, if the result of $\frac{|Ta-\ \ |}{Cr}$ was smaller or equal to one.

3. Results

3.1. Script A – Time reference extraction

Using the subset, described in chapter 2.3.1., led to the results depicted in Table 1. As shown in Table 1, the self-written Python script achieved superior numbers in all examined key figures.

Table 1. Results of the comparison of Script A and the parsedatetime library

	Script A	Parsedatetime library
Precision	99.6%	97.6%
Recall	96.4%	64.8%
Accuracy	98.0%	81.6%

3.2. Script B – Time calculation

Using the subset and method described in chapter 2.4.2., 223 out of the 296 reference notes were correctly classified. Therefore, Script B achieved an accuracy of 75.3%.

4. Discussion

Over the course of this paper, two different Python scripts were developed to enable a reliable time reference extraction from HF patients' clinical notes from the HerzMobil network. Script A was responsible for an extraction of time references and could surpass the paresedatetime library in all measured key figures. Script B subsequently used the output from Script A to calculate an interval, in which the corresponding time reference was expected to be. This algorithm worked with an accuracy of 75.3%.

The algorithms developed in Script A and Script B have only been tested within the environment of a HF telehealth network. The designed regular expressions were tailored to the notes and terms of the HerzMobil network. Therefore, by this study, it could not be proven, that Script A is a better date extraction software than the parsedatetime library. It has yet to be investigated if the results of this study apply to similar datasets as well. A test outside of the scope of the HerzMobil notes would be necessary to ensure a fair comparison to the parsedatetime library. Additionally, this could show, if Script A is also applicable for notes, which were derived from other sources than the HerzMobil network.

One reason for the superior results of Script A could be due to the testing phase, which has been done with notes from the HerzMobil network. During this phase the regular expressions were frequently altered and extended to cover the most common but also quite specific time references as well. Another reason could be that Script A solely focused on the German language, while the parsedatetime library implemented extraction algorithms for multiple languages.

Further investigating the reasons for false-positive results of the parsedatetime library was not possible, since

the result was only a calculated date without an exact reference to the underlying expression. Since the regular expressions were continuously adapted during the development process and therefore optimized for the HerzMobil notes, it could be possible that Script A is overfitted.

The calculation of specific dates and timespans works reasonably well with the algorithm of Script B. One issue that could be improved in future work is the distinction between a future and a past setting of the time reference. Although a list of indicators for either a future or a past setting should mitigate this problem, this method is not perfect and could use improvements.

As a next step, the two developed Python scripts could be implemented in the HerzMobil network either for retrospective analysis of all notes or, after slight adaptions, to support structured entry of data or for real time feedback in the live system. This feedback could be given in the form of automated calendar entries or automated creation and allocation of tasks. This would reduce manual work and, therefore, further improve the HerzMobil system.

Additionally, the underlying ideas and techniques of the scripts can be used to develop similar algorithms, which are not specifically designed to process HF related notes. In further research, both scripts could be slightly adapted and then be implemented in other telehealth systems, developed by the AIT.

5. Conclusion

We conclude that the HerzMobil network could be improved by implementing the two developed Python scripts. The developed scripts can reduce manual work and further improve the treatment process for patients. Therefore, this paper is a valuable contribution to the enhancement of the HerzMobil telehealth network and to similar telehealth solutions.

References

[1] P. Ponikowski, “2016 esc guidelines for the diagnosis and treatment of acute and chronic heart failure,” *European Heart Journal*, vol. 37, no. 27, pp. 2129-2200, Jul. 2016.

[2] A. Heidt et al, “HerzMobil Tirol network: Rationale for and design of a collaborative heart failure disease management program in Austria,” *Wiener klinische Wochenschrift*, vol. 126, Nov. 2014.

[3] F. Hebal et al, “Automated data extraction: merging clinical care with real-time cohort-specific research and quality improvement data.,” *J Pediatr Surg*, vol. 52, pp. 149-152, Jan 2017.

[4] E. Yehia et al, “Ontology-based clinical information extraction from physician’s free-text notes,” *Journal of Biomedical Informatics*, vol. 98, pp. 103276, 2019.

[5] R. Zhang et al, “Detecting clinically relevant new information in clinical notes across specialties and settings,” *BMC Med Inform Decis Mak*, vol. 17, Jul. 2017.

[6] H. J. Lee et al, “Identifying direct temporal relations between time and events from clinical notes,” *BMC Med Inform Decis Mak*, vol. 18, pp. 49, Jul. 2018.

[7] G. Gonzalez-Hernandez et al, “Capturing the patient's perspective: a review of advances in natural language processing of health-related text.,” *Yearb Med Inform*, vol. 26, pp. 214-227, Sep 2017.

[8] A. Eggerth et al, “Natural language processing for detecting medication-related notes in heart failure telehealth patients.,” *Studies in Health Technology and Informatics*, vol. 270, pp. 761-765, 2020.

[9] R. Sidhu and V. K. Prasanna, “Fast regular expression matching using FPGAs,” *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01),* pp. 227-238, 2007.

Address for correspondence:

Fabian Wiesmüller.
Reininghausstraße 13/1, 8020 Graz, Austria
fabian@wiesmueller.info