Oliver Wesely

+43 680 22 13 210 • oliver.wesely@gmx.at
https://github.com/olliwisly

Curriculum Vitae

Education

MSc in Software Engineering & Internet Computing **in progress**
Vienna University of Technology
Relevant Coursework *OOP, Algorithmics, Information Retrieval, Fixed-Parameter Algorithms and Complexity, Logics, Database Systems, Heuristic Optimization Techniques, Semantic Systems*

MSc in Biomedical Engineering **in progress**
Vienna University of Technology
Relevant Coursework *Medical Image Processing, Medical Physics of diagnostic imaging, Computerassisted Imaging Concepts, Biomedical Fluid Mechanics (selected Topics)*

MSc in Data Science and Machine Learning **2019**
University College London
GPA *71% (GB, Distinction)*
Relevant Coursework *Introduction to Machine Learning, Introduction to Statistical Data Science, Machine Vision, Graphical Models, Bioinformatics, Computational Modelling for Biomedical Imaging, Applied Machine Learning and Advanced Deep Learning and Reinforcement Learning (Deepmind)*
Thesis *"Electronic prescription: Using natural language processing to improve human computer interaction"*
Supervisor *Prof. Anthony Hunter and Dr. Jonathon Wong (UCLH)*

BSc in Economics **2017**
Vienna University of Business and Economics
GPA *3.5 (USA), 1.5 (Austria), Top 1% (student ranking)*
Thesis *"Bayesian Model averaging in economics"*

BSc in Statistics and Mathematics in Economics **2017**
Vienna University of Technology
GPA *3.3 (USA), 1.7 (Austria)*
Thesis *"How do naturally inspired search algorithms help to find the optimal trading strategy?"*

Professional Experience (Internships)

Data Science Internship at KPMG Vienna **2018**
Duration *1 month*
Experience *Conjunction of two hospital databases with a planned internship duration of 5 months. Due to disagreements with my project manager, ignoring various automation suggestions, I quit.*

Risk Management Internship at BMW Bank Munich **2017**
Duration *6 months*
Experience *Statistical analysis of big databases, implementation into a scorecard model; Testing of scorecard releases in ALM; Automation of monthly quality assurance in Oracle, SAS and VBA.*

Data Science Internship at Crédit Agricole CIB **2015**
Duration *3 months*
Experience *Automatize repetitive work in VBA (Excel with Bloomberg); Support in origination and sale of new bond issues; Analysis of target markets, business models, risk indicators and macro data.*

Compulsory Community Service

Paramedic at Red Cross (9 months) **2013**

Computer Skills

Advanced *Python (NumPy, TensorFlow, NLTK, Pandas, PyTorch, OpenCV, Matplotlib, ...), MATLAB*
Intermediate *R, Oracle, MS Office, LaTeX, Java*
Basic *Maple, Bloomberg, GAMS, SQL, PHP, SAS, C/C++*

Computer Skills

German *native speaker*
English *fluent (TOEFL: 7.5 – C1)*
Spanish *B2*
Russian *A2*

Hobbies

Running- marathon (3:30h), halfm. (1:30h)
Triathlon - half ironman (5:25h)
Cycling - 24h (624km)
Ballroom Dance - 4y dance instructor
Mountaineering

Oliver Wesely
Ph.D. Candidate

oliver.wesely@gmx.at
+43 680 22 13 210
Halbgasse 25, 1070 Vienna
https://github.com/olliwisly

Prof. Dr. Muhammad Shafique,
Prof. Dr Radu Grosu
Vienna University of Technology
Institute of Computer Engineering
Treitlstraße 3, 1040 Wien

Motivation Letter

September 18th, 2020

Dear Mr. Shafique and Mr. Grosu,

With this letter, I would like to express my interest in the PhD at the Cyber-Physical Research Division. I am particularly keen to apply for the doctoral program as the research focus of the project, investigate the applicability of machine learning techniques for healthcare applications, are an excellent match for my academic background.

Finishing the BSc in Statistics and Mathematics in Economics with a thesis about finding an optimal trading strategy using meta-heuristic optimization techniques, was my first encounter with a machine learning problem and changed my career plans profoundly. My interest in finding more powerful techniques resulted in my study of several machine learning books and attending my first Imaging lecture "Medical Image Processing" (Georg Langs, TU Vienna). This in turn spurred my application for the MSc in Data Science and Machine Learning at University College London, where I chose modules including Machine Vision (thought by Gabriel Brostow), Biomedical Imaging (by Daniel Alexander) and theoretical modules such as Graph Theory (by David Barber), Advanced Deep Learning and Reinforcement Learning (by Google DeepMind – David Silver, Hado van Hasselt,…). Set on acquiring a broad knowledge in machine learning, I wrote my master thesis in the field of NLP, supervised by Anthony Hunter. In order to improve my software engineering skills, I am currently enrolled in the corresponding MSc at TU, where I hope to get further in-depth knowledge within programming and specific topics of machine learning in my free time.

I now wish to continue my academic career with a PhD in computer science in your group, which has an outstanding expertise in various computational application in fields as Engineering, Physic and Biological Systems. I have read ongoing research related to the project description and recent articles about similar papers of scientists in your group.

Overall, I am profoundly convinced to have the commitment, creativity and ability to conduct independent scientific work and research throughout the PhD and within a multidisciplinary team. I wish to tackle the exhilarating, yet challenging path possibly leading to a research career and gain a lot of experience within a group of international scientists, building up a network and publish my research results at international conferences and journals.

After completing the PhD, I plan to pursue a postdoc placement within the field of AI and Machine Vision. In general, the rapid development of machine learning techniques and the abundantly available data are furthering fundamental changes to how scientific research is conducted in many scientific fields, from industrial to medical applications. The complex nature of these problems requires interdisciplinary approaches including machine learning, computer vision, medical knowledge and many more. Therefore, my research interest is not only in the mentioned fields above but also in various complex problems in cross-disciplinary sciences, as I am always motivated to acquire new knowledge and explore different fields of research.

Since I believe it is important to strike a sensible balance between academics and extracurricular activities, my personal interests are very diverse. Due to my passion for sport, I train for and have attended various triathlons, marathons and cycling races. Sport allows me to clear my mind and gives me a sense of calmness. Being an athlete demands discipline and dedication, qualities which help me to achieve my ambitions and goals.

Thank you for your consideration and I am looking forward to hearing from you soon.

Yours faithfully,
Oliver Wesely

Recommendation referees - contact information:

Anthony Hunter: anthony.hunter@ucl.ac.uk (supervisor master thesis)

Jonathon Wong: jonathon.wong1@nhs.net (supervisor master thesis) – Letter of recommendation appended

Re: Letter of recommendation

I am writing regarding Oliver Wesely in support of his application for his PhD application. I provided clinical supervision for his project in his Master of Science in data science and machine learning from 2018-2020.

We worked together as I provided clinical oversight and advice to his project as he wanted to have a project rooted in healthcare. Throughout his project Oliver was able to demonstrate proficiency in conceptualising, designing, data gathering and programming where he created a tool to assist in processing medical prescriptions leveraging machine learning techniques.

He demonstrated excellent communication skills and project management skills despite challenging conditions. Oliver was self-motivated and capable of troubleshooting efficiently and would no doubt be a valued addition to any team.

If there is any other information needed please feel free to contact me.

Kind regards,

Dr. Jonathon Wong, BSc MBBS MRCP PgCert
Anaesthesia and Intensive Care
Medical Fellow, Intensive Care Society
Intensive Care Fellow, King Edward VII Hospital

Email: Jonathon.wong1@nhs.net
Tel: 07515544293

Electronic Prescription: Using Natural Language Processing to improve Drug Prescription Processing

Oliver Wesely

supervised by

Anthony Hunter, UCL

Jonathon Wong, UCLH

This report submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Data Science and Machine Learning

Department of Computer Science
University College London

September, 2019

Abstract

In western countries, electronic prescription is considered to have a significant impact on the whole process of health-related data. It is replacing hand-written prescription, which has been the standard method for decades, by tackling the many difficulties of paper prescription, including saving time and costs, minimizing errors, reducing redundant paperwork and more. For instance, in Great Britain the public health care provider, the National Health Service (NHS), is currently in the final phase of implementing its renewed prescription processing, which is expected to save NHS about £300m within the first two years of use. However, current electronic prescribing systems are time consuming to learn and require intensive training for clinicians, even for the most basic prescriptions.

In my thesis, I develop different tools to support electronic prescription processing which should simplify the use, make prescribing fast, save time and money. The user interface will include only one input field for users to fill with any prescription related data, without any restrictions or limitations. The system will automatically classify the input into prescription related classes (allowing it to populate a standardized electronic prescription) while also highlighting severe interaction with any current patient medication. Due to confidentiality issues with patient specific data, I utilize the freely available data on BNF (British Nation Formulary) website, including all drug related information, to synthesise 10,000 prescriptions. For a prescription such as "Aspirin 300 mg every 4-6 hours IV starting 8am max per dose 600 mg" the system should be able to recognise and classify its components as: ['Drug Name', 'Dose', 'Unit', 'Frequency', 'Route', 'Starting Time', 'Maximum']. For such a classification I evaluate several NLP classification models, finding that the Averaged Perceptron Tagger model works best for the available data.

Dedication

A big thank you goes out to my dad, who made the whole master degree at UCL possible. He sadly passed away in July this summer and will never be able to read my master thesis. I am honoured, thankful and sure he would have been very proud. Therefore I dedicate my thesis to him.

Rest In Peace Dad!

Acknowledgement

Thanks to my supervisor Anthony Hunter, who was incredibly helpful and generous with his time, and my second supervisor Jonathon Wong who came up with the idea of the project and helped me with any medical background knowledge I needed regarding my data set. Without them this project would never have gone very far.

Finally thanks to my parents, without their help my postgraduate studies would have been impossible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For many years, hand-written prescriptions have been the standard method for doctors to record treatment decisions. Since the turn of the century, a shift towards digitalisation has revolutionized many industries. The medical field has resisted this trend, meaning that there are many opportunities for utilising technology to improve and advance patient treatment.

In particular, electronic prescriptions offer the chance to make a significant impact to the whole process of providing medical treatment, recording and studying health-related data. Broadly defined, an electronic prescription includes using any electronic device to enter, modify, review and generate or transmit a patient's prescription. It facilitates and ensures a safe transmission of prescription-related information between stakeholder of any kind, such as prescribers, dispensers, pharmacies, health plans and health insurers. The electronic prescription system "can overcome many problems of the paper prescribing process and will bring benefits including cost savings, reducing prescription errors, increasing prescription legibility, improving medication therapy outcomes, reducing redundant paperwork, electronically accessing to updated pharmacopeia information, and patient medication history." [1] Given these benefits, it has been discussed, tested, and implemented in

several European countries and the United States, with each implementation tailored to domestic needs.[1, 2, 3] However, many other countries, especially developing countries, still use hand-written manual prescription systems. Making electronic prescription standardized and accessible everywhere will bring benefits to health systems that need them the most. [1]

Great Britain is currently in the process of revolutionizing its prescription system. The 'Paperless 2020' initiative, published in 2018, expects all NHS providers to use electronic prescribing by 2020.[4] Additionally, the NHS 'Long Term Plan' for the next 10 years expects all providers to implement electronic prescribing in an effort to reduce errors by up to 30%. [5] The NHS recently began the final phase of implementing its renewed prescription processing system, which will be rolled out across the whole country by the end of 2020. After the nationwide roll-out of this new system, the NHS is expecting to save about £300m by 2021. These substantial savings are driven by reductions in the amount of time GPs and pharmacists require to write and dispense prescriptions using the new electronic system compared to the old system using paper processing. [4, 5]

1.1 Motivation

Current electronic prescribing systems are difficult to learn and require intensive training for every doctor, nurse and pharmacist. Even for the most basic prescriptions the process of writing and dispensing is very time consuming. The difficulty and complexity in using the software results in decreased efficiency while the need for new equipment increases costs. Current software requires a prescriber to navigate multiple screens, fields and drop-down menus in order to complete a single prescription, and this difficulty of use is a major disincentive for end users (doctors, nurses and pharmacists). The current drug prescription processes is tedious as the users have to select a drug first, then choose the route from a

drop-down menu, after which all prescription details have to be filled out in a combination of fields, drop-down menus and check boxes on a second screen, see appendix figures A.7a and A.7b. As a result these pieces of software lack real user experience and can be viewed as a setback for digitising medical prescriptions. The prospect of paying more for a system that adds a new layer of potential human error is unappealing. Therefore, a large body of clinical workforce prefers to continue with the traditional paper drug chart (which is often used as a fall-back method) due to its ease of use, simplicity and ubiquity.

1.2 Objectives

In this thesis I will discuss how machine learning can support electronic prescription processing to make prescribing faster and more user friendly for any stakeholder involved with medication prescriptions. Specifically, my goal is to build a prescription software which is simple to setup, and straightforward to use while requiring less hardware than current systems.

The main goal of the project is to improve the currently used prescription process by implementing a combination of two separate functions in a single user interface (UI). When users want to prescribe a drug, they would start typing into a single text box. Meanwhile, in the background, the first function, which will be called 'prediction' function, autocompletes the input to a list of possible prescriptions, using Natural Language Processing (NLP). To be able to provide such a suggestion list a trained Tagger model is used to classify the input portion, and predict the next word class and its corresponding word. Where the training data of the Tagger model is a set of web scraped and synthesised prescriptions from BNF's website. After the user has selected a suggestion or finished the free text input, it will be split up by the Tagger model into its corresponding class fields in the GUI. For instance, the Tagger model should be able to classify and split up a prescription shown

in (1.1) into its corresponding classes, where classes can be found beneath each word. However, in actual deployment the 'prediction' function would be trained on prescribers' historical prescription records and thus be tuned to each individual's unique prescription and writing style. The second function, called 'warning' function, uses the BNF database of known drug interactions to provide a user with warnings about any possible adverse reaction for a specific patient, knowing a patient's current medication list. If the level of severity of the interaction is sufficiently high, the corresponding drug will be highlighted in the suggestion list and a warning pops up showing relevant information, including the drugs the interaction risk is coming from and its level of severity.

Overall, the list of requirements I want to incorporate in my system include:

- Synthesising a data set of reasonable prescriptions
- Finding an accurate prescription classification model
- Implementing a prescription suggestion list based on user input
- Setting up a 'warning' - highlighting severe drug interactions
- Simplifying the overall prescription process for end users

The aim of the thesis is to solve all these requirements. In Chapter 3 I am trying to implement methods to address all these requirements. After that, I am evaluating all classification model in Chapter 4, to find the most accurate one. Whereas in Chapter 5 I will combine and show all these requirements within a user interface, especially to demonstrate the improvement of the drug prescription process as my last point of the list requires.

1.3 Data

To prevent confidentiality issues with the data set, I will not use patient specific data. Instead I have relied on the open drug data as published on the BNF website. BNF is UKs pharmaceutical reference book which contains commonly used medicines in the UK including information and advice on prescription, published on the website of the National Institute for Health and Care Excellence (NICE, see https://www.nice.org.uk/). It provides information about indications, doses, side effects and interactions for over 70,000 medicines. Without loss of generality, I will narrow down the list of drugs, I will use in my research, to all drugs starting with letter "A". [6]

To get reasonable training data, I web scraped BNF's website to synthesize mock free text prescriptions. To generate this wide-ranging, comprehensive data set I used regular expressions to identify words from nine different classes that together form a complete prescription, which will be extracted from the 'Indication and Dose' information on BNF. The nine most commonly used and important classes of words in a drug prescription are those shown in table 1.1 including class instances.

Class	**Abb.**	**Examples**
Drug Name	DN	"Aspirin", "Amoxicillin", "Asparagine", "Abacavir",...
Dose	DO	1, 5, 100, 500, ... (mostly integer numbers)
Unit	UN	"mg", "g", "drop", "tablet", "capsule", ...
Frequency	FR	"once daily (o.d.)", "every 8 hours", "three times a day (t.d.s.)", ...
Route	RO	"PO", "IN", "IV", "SC", ...
Starting Time	ST	"start p.r.n", "start 1 pm", "starting as required", ...
Duration	DU	"3 doses", "5 days", "2-3 weeks", "3 months", ...
Maximum	MA	"max 12 g per day", "max 7 doses per week", "max 7 days", ...
Minimum	MI	"min 2 hours", "min 6 hours", ...

Table 1.1: Classes and corresponding instances

After extracting available class instances, they will be randomly ordered and noise will be added, to simulate different user typing styles and errors, and to generate reasonable

prescriptions. Three such example prescriptions can be found in (1.1), (1.2) and (1.3) including all corresponding word classes beneath. All medical abbreviations used for the prescriptions are explained in Appendix B.

$$\underbrace{Aspirin}_{\text{'Drug Name'}}\ \underbrace{300}_{\text{'Dose'}}\ \underbrace{mg}_{\text{'Unit'}}\ \underbrace{every\ 4-6\ hours}_{\text{'Frequency'}}\ \underbrace{IV}_{\text{'Route'}}\ \underbrace{Starting\ 8am}_{\text{'Starting Time'}}\ \underbrace{max\ per\ dose\ 600\ mg}_{\text{'Maximum'}} \tag{1.1}$$

$$\underbrace{500}_{\text{'Dose'}}\ \underbrace{micrograms}_{\text{'Unit'}}\ \underbrace{tid}_{\text{'Frequency'}}\ \underbrace{alprazolam}_{\text{'Drug Name'}}\ \underbrace{start\ prn}_{\text{'Starting time'}}\ \underbrace{po}_{\text{'Route'}}\ \underbrace{3\ days}_{\text{'Duration'}} \tag{1.2}$$

$$\underbrace{starting\ 8am}_{\text{'Starting Time'}}\ \underbrace{po}_{\text{'Route'}}\ \underbrace{od}_{\text{'Frequency'}}\ \underbrace{75}_{\text{'Dose'}}\ \underbrace{mg}_{\text{'Unit'}}\ \underbrace{asparagine}_{\text{'Drug Name'}}\ \underbrace{min\ 6\ hours}_{\text{'Minimum'}}\ \underbrace{3-9\ days}_{\text{'Duration'}} \tag{1.3}$$

My thesis is organised as follows: In Chapter 2, I provide a literature review covering research, which already has been done on related topics. Additionally, in this section I provide theoretical background knowledge (especially relating to Machine Learning) needed for subsequent chapters. Chapter 3 will initialise and explain all methods used in later chapters. In Chapter 4 I will evaluate the best Tagger model, which will be used as my classification model in the following chapter, Chapter 5, which provides an explanation of how I am improving the drug prescribing process. After that, I am using an experimental data setup to retrain the Tagger model and analyse its results in Chapter 6. My thesis will end with some conclusions in Chapter 7 and a subsequent appendix including screenshots of the BNF website, the old user interface and the new developed interface, along with a list of medical abbreviations and the online repository of my Python Code.

Chapter 2

Literature Review & Theoretical Background

2.1 Related Work

In recent years there has been substantial amount of research in the field of electronic prescription. Multiple general papers have compared advantages and drawbacks of the new and the old system, I will highlight a few in this abstract. One the one hand a paper says "There are many problems with the paper prescription system which, according to studies have jeopardized patients' safety and negatively affected the outcomes of medication therapy" [7], whereas another identifies "three major themes that may increase the potential for medication errors with e-prescribing ... (1) increased cognitive burden on pharmacy staff, ... (2) interruptions during the e-prescription dispensing process, and (3) communication issues with prescribers, patients, and among pharmacy stuff." [8]. Overall "there is a lack of consensus within the literature on the impact of electronic prescription ... future research should explore the strategies resulting in a positive impact" [9] after analysing 171 papers throughout different journals which explored the effect of electronic prescription

and "identified four key themes: communication, time taken to complete tasks, clinical workflow and workaround" [9]. Another one used 34 publications finding 594 elements as barriers or facilitators to electronic prescription including that electronic prescribing "was facilitated by design and technical concerns, interoperability, content appropriate for the users, attitude towards e-prescribing, productivity, and available resources." [10] and they also show that "the same factor can be seen as a barrier or a facilitator depending on the project's own circumstances." [10]

While the previous papers explore the challenges associated with implementing electronic systems, other papers describe how the use of NLP can tackle different problems related to electronic prescription. For example in paper [11] medications and related information, including medication name, dosage, mode, frequency, duration and reason for drug administration, will be extracted from institutions' clinical text using different NLP tools. Paper [12] uses NLP to identify and minimize patients' risk of adverse events, addressing the problems caused by the majority of drug interaction data being stored as free text. And finally, in [13] they also use prescription guidelines offered by UK NICE BNF as I will do, and attempts to interpret its semantic meanings using NLP in order to provide prescription error checking in an IT based clinical decision support(CDS). [13]

In conclusion, there are several papers documenting the relative benefits of electronic vs paper systems, there is still a lack of consensus regarding the impact of electronic prescription, but they generally agree that future research should explore strategies which have a positive impact. Additionally, several papers have used NLP to investigate methods of predicting prescriptions. Likewise, previous work has attempted to provide drug interaction warnings. However, none have combined these two techniques with an emphasis on simplicity both in terms of end user use and hardware requirements.

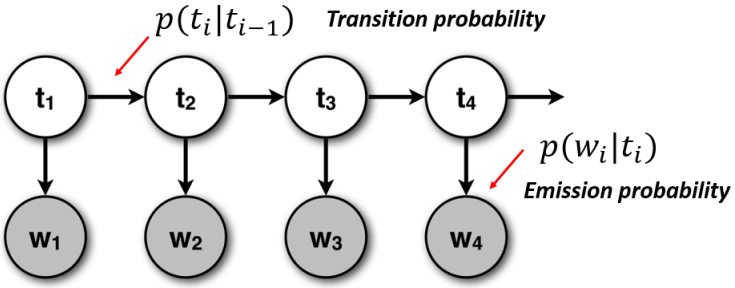


Figure 2.1: HMM transition and emission probabilities [14]

2.2 Part-of-Speech Tagger

2.2.1 HMM Tagger

The Hidden Markov Model is a probabilistic sequence model which is computing the probability distribution of possible sequences of labels to be able to choose the most likely one.

A HMM is based on the Markov chain, which is a model containing probabilities of sequences of random variables, states, which can have values from a set, for example a set of words. A Markov chain makes a very strong assumption, the so called Markov assumption, which is predicting the future of a sequence relying only on the current state and all states in the past have no direct impact except via the current state. Consider a sequence of state variables $q_1, q_2, \ldots, q_i$, than the Markov assumption would be:

$$P(q_i|q_1, \ldots, q_{i-1}) = P(q_i|q_{i-1}) \tag{2.1}$$

Hidden Markov Model

In many cases events cannot be observed directly, the so called hidden events. For instance, part-of-speech tags will not be observed in a text normally, they rather have to be inferred from the word sequence, therefore I call them hidden. A Hidden Markov model (HMM)

combines both, observed events (words) and hidden events (part-of-speech tags), and is specified by the following components:

$Q = q_1, q_2, \ldots, q_N$	a set of N states
$A = a_{1,1}, \ldots, a_{i,j}, \ldots, a_{N,N}$	a transition probability matrix A, $a_{i,j}$ is the transition probability from state i to state j, s.t. $\sum_{j=1}^{N} a_{i,j} = 1 \forall i$
$O = o_1, o_2, \ldots, o_T$	a sequence of T observations, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of observation likelihoods, called emission probabilities, the probability of an observation o_t being generated from a state i.
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an initial probability distribution over all states, where π_i is the probability that the Markov chain starts in state i. ($\sum_{i=1}^{n} \pi_i = 1$)

A first-order HMM has two simplifying assumptions, the first is the Markov Assumption, equation 2.1, the same as for any first-order Markov chain. The second assumption is called the output independence, meaning that the probability of an output observation o_i only depends on the state q_i that produced the observation:

$$P(o_i|q_1, \ldots, q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i|q_i) \tag{2.2}$$

HMM Tagger components

A HMM consist of a transition matrix A, containing all transition probabilities $P(t_i|t_{i-1})$, and the emission matrix B, consisting of emission probabilities $P(w_i|t_i)$, see Figure 2.1. The maximum likelihood estimate (MLE) of the transition probabilities can be calculated by counting, function $C()$, the times how often the second specific tag follows the first specific tag, $C(t_{i-1}, t_i)$, out of all occurrences of the first specific tag, $C(t_{i-1})$, and therefore:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \tag{2.3}$$

Similarly the MLE of the emission probabilities in matrix B is given by:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)} \tag{2.4}$$

Decoding HMM - The Viterbi algorithm

Determining the hidden variables corresponding to the sequence of observations of any HMM is called decoding. For part-of-speech tagging the goal is to find the most probable tag sequence t_1^n for a given observation sequence w_1^n using Bayes' rule:

$$\begin{aligned} \hat{t}_1^n &= \underset{t_1^n}{\text{argmax}}(P(t_1^n|w_1^n)) \\ &= \underset{t_1^n}{\text{argmax}}(\frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}) \\ &= \underset{t_1^n}{\text{argmax}}(P(w_1^n|t_1^n)P(t_1^n)) \end{aligned} \tag{2.5}$$

Taking both HMM assumptions into account:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \tag{2.6}$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \tag{2.7}$$

Plugging equation 2.6 and 2.7 in 2.5, results in:

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}}(P(w_1^n|t_1^n)P(t_1^n)) \approx \underset{t_1^n}{\text{argmax}}(\prod_{t=1}^{n} \overbrace{P(w_i|t_i)}^{emission} \overbrace{P(t_i|t_{i-1})}^{transition}) \tag{2.8}$$

The commonly used decoding algorithm for HMMs is the Viterbi algorithm as shown in algorithm 1.

[15]

Algorithm 1: The Viterbi Algorithm [16]

Result:

Step 1: Initialization

for *each state s from 1 to N do* **do**

$viterbi[s, 1] = \pi_s b_s(o_1)$

$backpointer[s, 1] = 0$

end

Step 2: Induction

for *each time step t from 2 to T* **do**

for *each state s from 1 to N* **do**

$viterbi[s, t] = \max_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

$backpointer[s, t] = \operatorname{argmax}_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

end

end

Step 3: Termination

$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s, T]$

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} backpointer[s, T]$

Step 4: Backtracking

for *each time step t from T-1 to 1* **do**

$bestpathpointer[t] \leftarrow backpointer[bestpathpointer[t + 1], t + 1]$

end

$bestpath \leftarrow (bestpathpointer[1], \ldots, bestpathpointer[T])$

return *bestpath,bestpathprob*

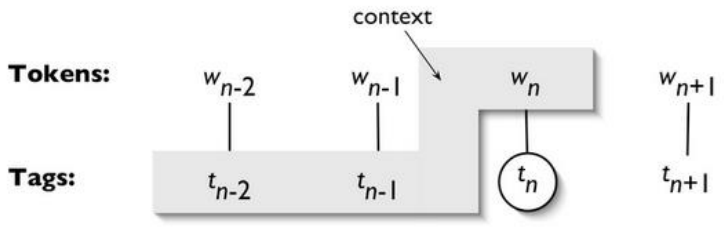


Figure 2.2: Tagger Context [17]

2.2.2 *N*-gram Tagger

A N-gram is a sequence of N words, which I use to estimate the probability of the last word of the N-gram given the previous words. A 1-gram is a one-word sequence and is called Unigram, a 2-gram will be called Bigram and a 3-gram Trigram. The probability of a word w given its history h, $P(w|h)$, can be estimated by relative frequency counts, but to include all available history would be computationally expensive, therefore I use N-gram models which instead approximate the history by just the last N words. The general approximation of any N-gram model for the conditional probability of the next word is:

$$P(w_n|w_1, \ldots, w_{n-1}) \approx P(w_n|w_{n-N+1}, \ldots, w_{n-1}) \tag{2.9}$$

In the Bigram model (N=2) this would result in the Markov assumption, I already have seen in the last section. The example shown in Figure 2.2 shows a Trigram model (N=3), where I consider the two preceding tags, t_{n-2} and t_{n-1}, to predict the tag t_n of the current word w_n.

To be able to get the probability of a complete word sequence I apply the chain rule of probabilities and substitute equation 2.9:

$$
\begin{aligned}
P(w_1, \ldots, w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \ldots P(w_n|w_1, \ldots, w_{n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_1, \ldots, w_{k-1}) \\
&\approx \prod_{k=1}^{n} P(w_k|w_{k-1})
\end{aligned}
\tag{2.10}
$$

To estimate these probabilities I use MLE. I get the MLE estimate by counting different values in the N-gram sequences. For instance, if having a Bigram (w_{n-1}, w_n) I would count the number of all Bigrams that have both words and divide it by all Bigrams with the same first word w_{n-1} and get:

$$
P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{\sum_w C(w_{n-1}, w)} \tag{2.11}
$$

As the sum of all Bigram counts, that start with w_{n-1}, must be equal to the Unigram count, I can simplify equation 2.13:

$$
P(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})} \tag{2.12}
$$

And the generalised MLE N-gram parameter estimation form is:

$$
P(w_n|w_{n-N+1}, \ldots, w_{n-1}) = \frac{C(w_{n-N+1}, \ldots, w_{n-1}, w_n)}{C(w_{n-N+1}, \ldots, w_{n-1})} \tag{2.13}
$$

A practical issue which has to be mentioned at this point is that I need to augment each prescription with a special string "$< s >$" N-1 times at the beginning of the sentence and a special end-symbol "$< /s >$" at the end of a sentence. The special symbols at the beginning give us the N-gram context of the first word and the one at the end makes the N-gram grammar a true probability distribution, without it the prescription probabilities

for all prescriptions of a given length would sum to one and therefore I would have an theoretically infinite set of probability distributions with one distribution per prescription length. [15]

Backoff Interpolation

If there is no example of a particular N-gram, for instance a Trigram w_{n-2}, w_{n-1}, w_n and I cannot compute $P(w_n|w_{n-1}, w_{n-2})$, in such a situation the Bigram will be used as Backoff and instead try to estimate the probability $P(w_n|w_{n-1})$, similarly if there are no particular Bigrams as such, the Unigram is the second Backoff $P(w_n)$. The last and final Backoff can be a default tagger for example, which is setting all remaining tags to a specific class. In other words, Backoff to a lower-order N-gram model will take place if there is zero evidence for a higher order N-gram. [15]

In contrast if I use an interpolation I mix all probability estimates from all N-gram estimators by weighing and combining all counts. The simplest version is the linear interpolation, where all models are linearly interpolated, and the N-gram probability estimated. For instance, if I consider the Trigram probability $P(w_n|w_{n-2}, w_{n-1})$, it can be estimated by:

$$\begin{aligned} \hat{P}(w_n|w_{n-2}, w_{n-1}) = \quad & \lambda_1 P(w_n|w_{n-2}, w_{n-1}) \\ & + \lambda_2 P(w_n|w_{n-1}) \\ & + \lambda_3 P(w_n) \end{aligned} \tag{2.14}$$

such that $\sum_i \lambda_i = 1$. [15]

2.2.3 Averaged Perceptron Tagger

Perceptron algorithm

The Perceptron algorithm I am using is an implementation from Honnibal Matthew ([18]) which is based on an older version of the Averaged Perceptron described by Collins Michael ([19]). Collins' algorithm uses features, such that it behaves like an HMM tagger and standard Viterbi decoding is possible. All possible tags are generated for any given position. Transitions are based on binary-valued features and output scores on the weights of the candidate tags. A feature set is based on feature templates which shows concrete patterns and values from the training data. The feature templates I am using during training can be found in table 2.1, where w_i is the current word and $(w_1, \ldots, w_n)$ the entire sentence, respectively for tags $(t_1, \ldots, t_i, \ldots, t_n)$.

Bias	(acts like a prior)	Current word	w_i
Previous word	w_{i-1}	Two words back	w_{i-2}
Next word	w_{i+1}	Two words ahead	w_{i+2}
Previous Tag	t_{i-1}	Two tags back	t_{i-2}
Previous Two Tags	t_{i-1}, t_{i-2}	Previous Tag and current word	t_{i-1}, w_i
Current word prefix - length 1	w_{i_0}	Current word suffix - length 3	$\mathrm{w}_{i_{-3:}}$
Previous word suffix - length 3	$\mathrm{w}_{i-1_{-3:}}$	Next word suffix - length 3	$\mathrm{w}_{i+1_{-3:}}$

Table 2.1: Feature templates

During the Averaged Perceptron algorithm weights will be iteratively determined while training the algorithm. Binary-valued features describe the tag being predicted and its context, and will be derived from any available information of the text at the point of decision. Depending on the context, features are true or false, whereas true features lead to the fact that they describe a tag and its context. The Perceptron algorithm keeps weight coefficients for all features which might change at every training prescription and after its final update they will be stored. After the training, all coefficients of true features in a

given context will be summed up and will be passed to the Viterbi algorithm as a transition and output weight for the current state. The mathematical expression for that:

$$w(C,T) = \sum_{i=1}^{n} \alpha_i \phi_i(C,T), \tag{2.15}$$

with $w(C,T)$ being the transition weight for tag T in context C, the weight coefficient α_i of the i^{th} feature and the features evaluation for context C and tag T $\phi_i(C,T)$, for n features. In the Averaged Perceptron algorithm the values of every coefficient are summed up after each update and will be divided by the number of iterations after the last update to get the arithmetic average. Averaging makes the algorithm resistant to weight oscillation and it will pay less attention to the few examples it is getting wrong and overall it will lead to an improvement of its performance. [18, 20]

2.3 *k*-Fold Cross Validation

In k-fold cross-validation the data is split-up in k, equally sized, folds. Each fold should be a good representative of the whole data set, therefore a process of rearranging the data, called stratification, is commonly applied prior to the split-up. k iterations of training and validation are performed, within each iteration a different fold will be used as the validation set (test set) and the k-1 remaining folds as the training set, graphically shown in Figure 2.3. Every performance metric will be evaluated at every iteration step, and after finishing all iterations the final performance metrics are the averages of all respectively evaluated values.

There are multiple factors to consider when choosing an appropriate value for k. A larger k would be desirable, as there would be more performance estimates and the training set size is closer to the full data size. However, with an increasing value of k the size of the test set is shrinking and therefore less precise and less fine-grained measurements of the

performance metrics. Within the data mining community it seems to be that $k = 10$ is a good compromise. [21]

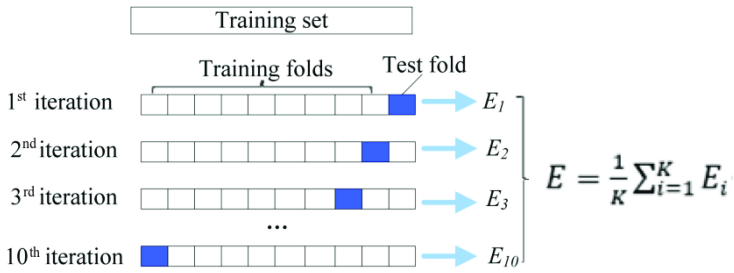


Figure 2.3: 10-fold Cross Validation[22]

2.4 Performance Measurements for the Comparison of Classifiers

The goal is to find a classifier that works best for the previously outlined multi-class task, therefore I need to use performance measurements tailored to this setting. These performance measurements include the Confusion Matrix, Average Accuracy, Error Rate, Precision, Recall and F-Score.

2.4.1 Confusion Matrix

For estimating the performance of a multi-class classifier the confusion matrix is a commonly used tool. The probabilistic definition of the true confusion matrix and the empirical confusion matrix are defined as:

DEFINITION 1 (True and empirical confusion matrix) *The true confusion matrix* $\boldsymbol{A} = (a_{l,j}) \in \mathbb{R}^{K \times K}$ *of a classifier* $h : X \leftrightarrow \{1, \ldots, K\}$ *over a distribution* D *is defined as follows:*

$$\forall l, j \in \{1, \ldots, K\}, \boldsymbol{a}_{l,j} \stackrel{\text{def}}{=} \mathbb{E}_{x|y=l}\mathbb{I}(h(\boldsymbol{x}) = j) = \mathbb{P}_{(\boldsymbol{x},y)\sim D}(h(\boldsymbol{x}) = j|y = l).$$

For a given classifier h and a sample $S = \{(x_i, y_i)\}_{i=1}^m \sim D$, the empirical confusion matrix $\boldsymbol{A}_S = (\hat{a}_{l,j}) \in \mathbb{R}^{K \times K}$ of h is defined as:

$$\forall l, j \in \{1, \ldots, K\}, \hat{\boldsymbol{a}}_{l,j} \stackrel{\text{def}}{=} \sum_{i=1}^{m} \frac{1}{m_l} \mathbb{I}(h(\boldsymbol{x}_i) = j) \mathbb{I}(y_i = l).$$

All rows of these matrices sum up to 1, the diagonal entries represent the capability of the classifier h to be able to classify correct, whereas all non-diagonal entries correspond to any mistakes of the classifier. [23]

2.4.2 Selected Measurements

Consider a problem of estimating k classes for a test set containing n instances. True classes are noted C_i and predicted classes are noted $\hat{C}_j$. Measures are mostly built from results of the confusion matrix and not from raw data, therefore I evaluate a confusion matrix as can be found in table 2.2, where terms $n_{i,j}$ correspond to instances classified in class j by the classifier when they actually belong to class i. For some measurements proportions are needed rather than the counts, defined by $p_{i,j} = n_{i,j}/n$, and the sum over row i and column j are respectively noted by $n_{i,+}$ and $n_{+,j}$. Similar definitions exist for proportions: $p_{i,+}$ and $p_{+,j}$.

	$\hat{C}_1$	$\hat{C}_2$	$\ldots$	$\hat{C}_k$
C_1	$n_{1,1}$	$n_{1,2}$	$\ldots$	$n_{1,k}$
C_2	$n_{2,1}$	$n_{2,2}$	$\ldots$	$n_{2,k}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
C_k	$n_{k,1}$	$n_{k,2}$	$\ldots$	$n_{k,k}$

Table 2.2: General Confusion Matrix

Looking at one particular class i, it can be distinguished between four different types of evaluating the classifier, see table 2.3:

Type	Explained	Formula
True positives (TP)	Correctly classified as $\hat{C}_i$	$n^i_{TP} = n_{i,i}$
False positives (FP)	Incorrectly classified as $\hat{C}_i$	$n^i_{FP} = n_{+,i} - n_{i,i}$
False negatives (FN)	Incorrectly not classified as $\hat{C}_i$	$n^i_{FN} = n_{i,+} - n_{i,i}$
True negatives (TN)	Correctly not classified as $\hat{C}_i$	$n^i_{TN} = n - n_{TP} - n_{FP} - n_{FN}$

Table 2.3: Classification Types

Overall Success Rate

One of the most popular measurement for classification accuracy is the overall success rate:

$$OSR = \frac{1}{n} \sum_{i=1}^{k} n_{i,i}, \tag{2.16}$$

which is the trace of the confusion matrix divided by n, which is multi-class symmetrical and corresponds to the observed proportion of correctly classified instances.

Marginal Rates

There exists a number of class specific measurements, including True positive rate (TPR, sensitivity, recall) and True negative rate (TNR, specificity), both reference-oriented as both consider confusion matrix rows. TPR is the proportion of class instances classified in the same class, respectively TNR of other class instances classified in another classes.

$$TPR_i = n^i_{TP}/(n^i_{TP} + n^i_{FN}) \tag{2.17}$$

$$TNR_i = n^i_{TN}/(n^i_{TN} + n^i_{FP}) \tag{2.18}$$

On the other hand the Positive Predictive Value (PPV, precision) and the Negative Predictive Value (NPV) are estimation oriented measures and consider the confusion matrix columns. PPV is the proportion of instances predicted to be from the considered class and indeed are, respectively NPV.

$$PPV_i = n^i_{TP}/(n^i_{TP} + n^i_{FP}) \tag{2.19}$$

$$NPV_i = n^i_{TN}/(n^i_{TN} + n^i_{FN}) \tag{2.20}$$

All four measurement range from 0 to 1.

Another class specific and symmetric measurement is called F-measure and corresponds to the harmonic mean of PPV and TPR, interpreted as a measure of overlapping between the true and estimated classes (0 means no overlap, 1 complete overlap).

$$F_i = 2 * \frac{PPV_i \times TPR_i}{PPV_i + TPR_i} = \frac{2n^1_{TP}}{2n^i_{TP} + n^i_{FN} + n^1_{FP}} \tag{2.21}$$

[24]

Comparison Metrics

The quality of overall classifications will be assessed with macro averaging, meaning that the general measure is the average of the same measures calculated for all classes. All classes will be treated equally when using macro averaging.

In the following list of equations an average accuracy (AAcc) is the average per-class effectiveness of a classifier and the error rate (ER) is the average per-class classification error. These and other measures such as Precision (P), Recall (R) and F-Score will be defined in the following:

$$AAcc = \frac{1}{k}\sum_{i=1}^{k} \frac{n^i_{TP} + n^i_{TN}}{n^i_{TP} + n^i_{TN} + n^i_{FP} + n^i_{FN}} \tag{2.22}$$

$$ER = \frac{1}{k}\sum_{i=1}^{k} \frac{n^i_{FP} + n^i_{FN}}{n^i_{TP} + n^i_{TN} + n^i_{FP} + n^i_{FN}} \tag{2.23}$$

$$P = \frac{1}{k} \sum_{i=1}^{k} PPV_i \tag{2.24}$$

$$R = \frac{1}{k} \sum_{i=1}^{k} TPR_i \tag{2.25}$$

$$F - Score = 2 * \frac{P \times R}{P + R} \tag{2.26}$$

[25]

Chapter 3

Methods

In this Chapter I will implement all the methods needed for my prescription process. After that, the Tagger models will be evaluated in the following Chapter and in Chapter 5 the prescription process will be developed, including all methods explained, within a graphical user interface. To see how the methods and the prescription process should work in the user interface, I created a model architecture of the process in Figure 3.1.

At first the process asks for a prescription input. As soon as the input length is at least 3 characters the 'prediction' function in the will be activated and the process will try to generate possible suggestions and a correction, details can be found in section 3.3. If the suggestion list has at least one entry and assuming the 'warning' function is inactive, the suggestion list will be displayed in a drop-down menu. If the user selects any of these suggestions it will be inserted into the 'Prescription' input field and the suggestion list will be deleted. If the user is changing the input the 'prediction' function process will be repeated with the new input, but if the user wants to split up the current prescription the 'Split Up' button has to be pressed, followed by the classification using the best evaluated tagger model. The considered tagger models are going to be evaluated in chapter 4 and already have been implemented in the last chapter in section 2.2. If the classification is

'valid', which will be explained in chapter 5, the classified words will be inserted to the corresponding class fields of the user interface.

In the case of an active 'warning' function, which the user can activate by ticking the 'Interaction' check box, a current medication will be generated for demonstration reasons. That would be the patient specific medication in a real deployment. The activation of the 'warning' function will highlight any suggestions including drugs which have a risk of interaction with any current medication, as will be defined in section 3.1.4 and implemented in chapter 5. In a later step of the prescription process, after getting a 'valid' input classification, the activation of this function would additionally lead to a pop up message in the user interface, including all interaction details.

But to be able to use the drug prescription process as such, data has to be synthesised first, which will be explained in the following section, section 3.1.

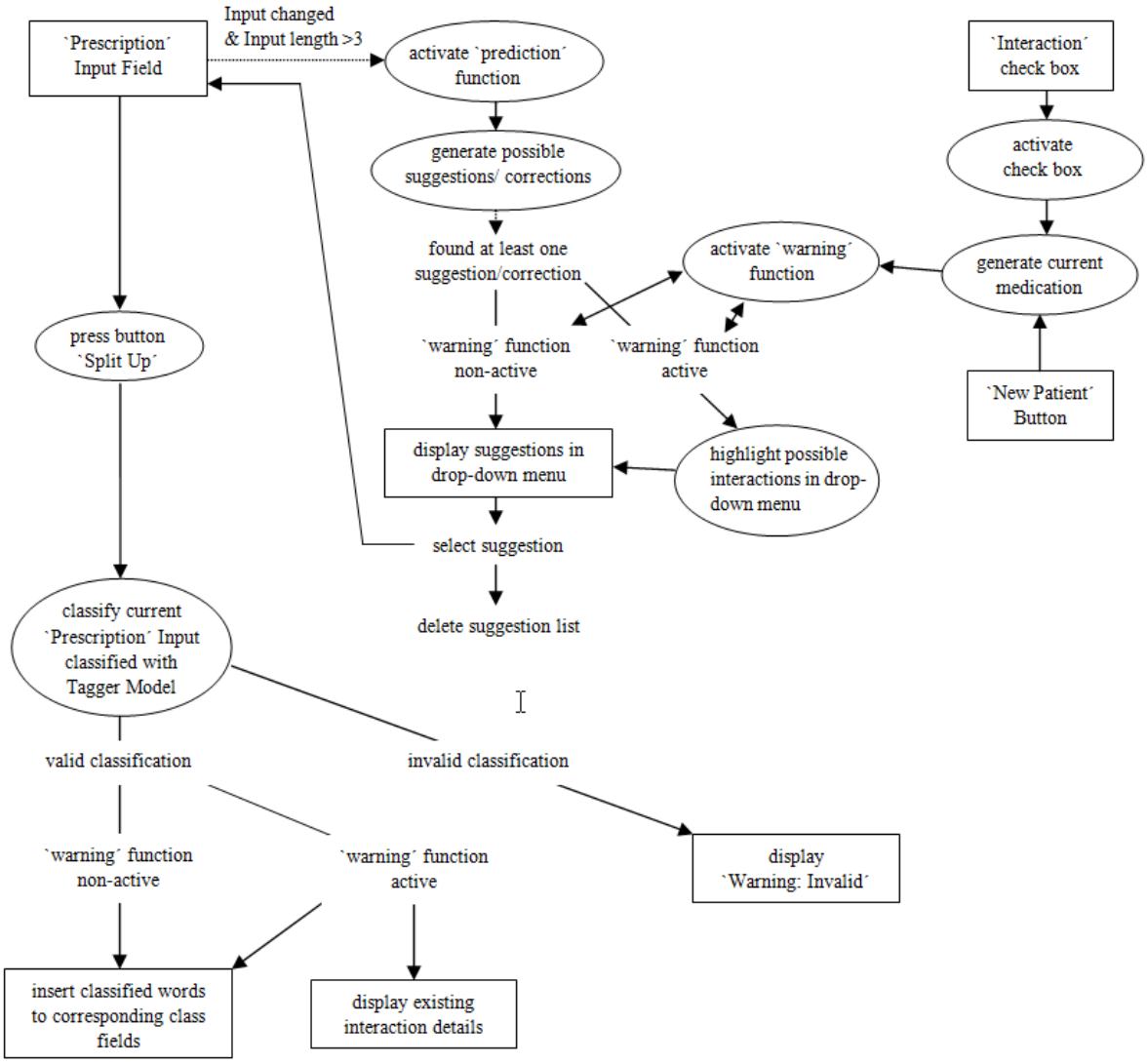


Figure 3.1: Drug Prediction Process Architecture

3.1 Synthesise Data

At first I had to synthesise data from the BNF's website, which includes 132 different drugs starting with letter 'A'. To get a feeling of what it means to scrape all the relevant information, I added a screenshot of BNF's Drug List in Figure 3.3a and the drug Abacavir as an example in Figure 3.3b, to demonstrate what the British National Formulary looks like. In Figure 3.3b web scraping all useful data is easy, since there is only one dose:

$$\text{“}600\ mg\ daily\ in\ 1-2\ divided\ doses\text{”} \tag{3.1}$$

which would lead to the data point: “Abacavir” [‘Drug Name’] - “by mouth” [‘Route’] - “600” [‘Dosage’] - “mg” [‘Unit’] - “daily in 1-2 divided doses” [‘Frequency’]. However, this is a rather rare case and normally it has multiple indications with multiple doses.

3.1.1 Data Assumptions

Without loss of generality, I will make some assumptions about the synthesised data at this point. These assumptions allow the method to focus on the more general cases and ignore all rare, special cases. For instance, looking at drug Amoxicillin in Figure A.1 it can be seen that the doses can vary depending on the age of a patient, which leads to my first assumption that I only use doses for adults. Applying this assumption would lead to three remaining data points for this specific indication, but there exist 20 more indications for Amoxicillin. The drug Alprostadil is even more complicated as most of the doses include changing dosages, as seen in Figure A.2, hence my next assumption will set a maximum length of the dose string. In Figure 3.2, one can see a box plot of all string lengths with a mean length of 75, the first quartile (25^{th} percentile) of 29.25, the third quartile (75^{th} percentile) 154.5, leading to an interquartile range of 125.25. The lower whiskers is the shortest dose string with length 5 and for the upper whiskers the maximum value is

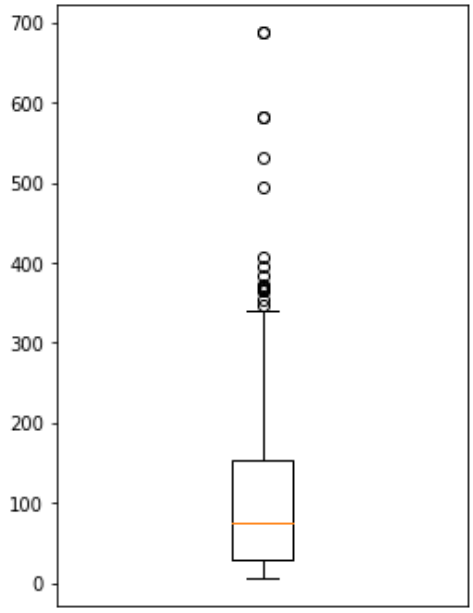


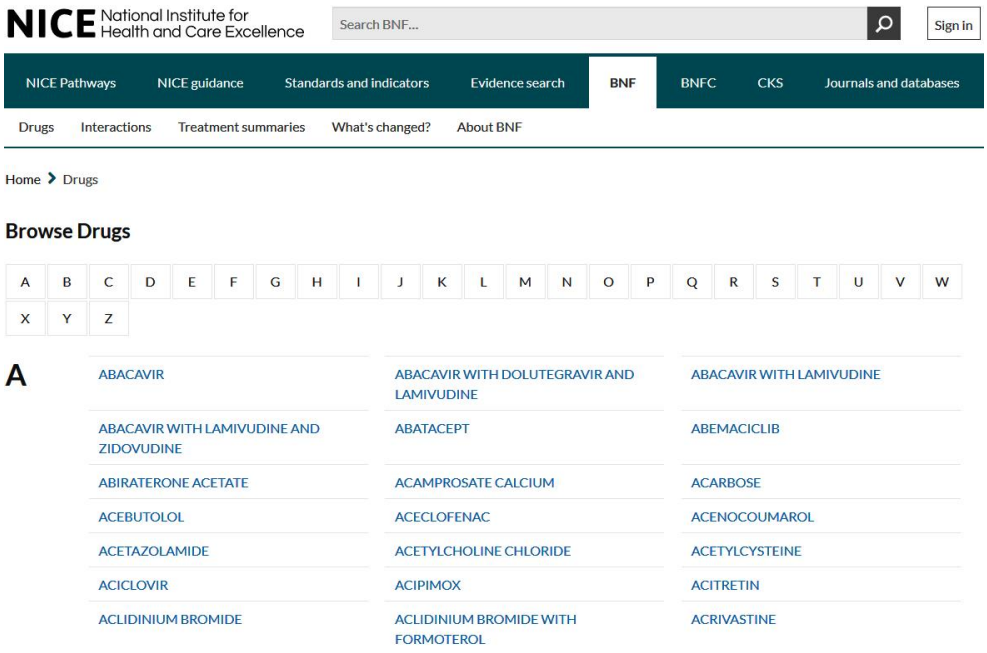
Figure 3.2: Boxplot - Dose string length

$154.5 + 125.25 * 1.5 = 342.375$, therefore is set at 340. All values outside the whiskers are considered to be outliers, therefore I will remove all doses with string length above 340. As a further example I included the entire list of indications and doses of Aspirin in Figure A.3, A.4 and A.5, which I will refer to in a later stage of my analysis.

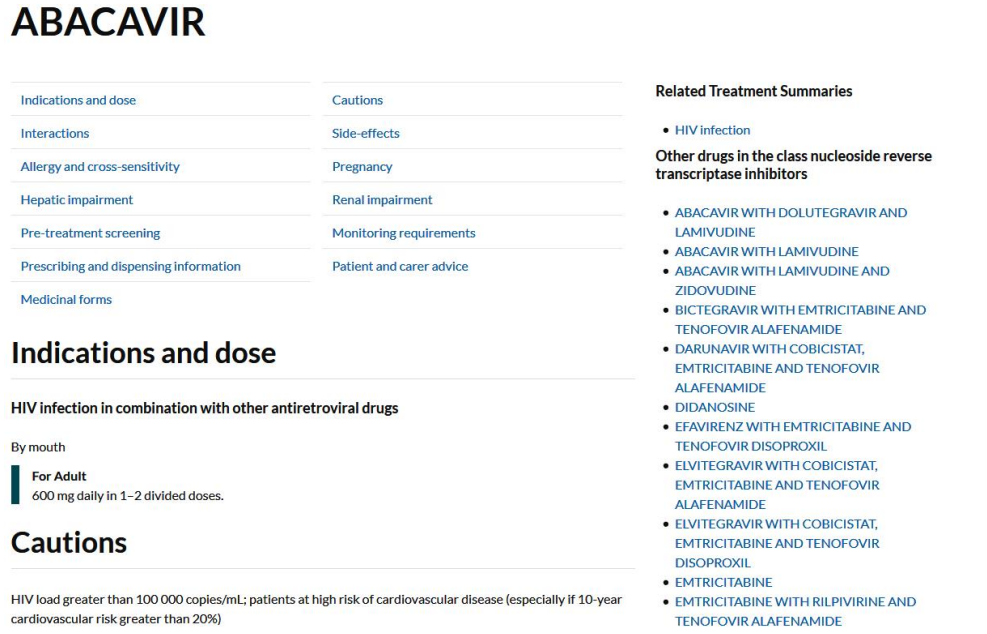
The final data assumption will involve handling uninformative instructions due to a lack of information. For instance, some doses include terms as "consult product literature", "consult local protocol" and "consult expert dental sources", meaning there is not enough information in the dose and hence these data points will not be included in the synthesised data set.

3.1.2 Extract information

Figure 3.3b shows the drug route can be easily scraped from a html file - in this case the route would be "by mouth" for which the corresponding medical abbreviation is "PO". All possible other classes can be found in the dose itself, next to the title "For Adult", see



(a) BNF Drug List



(b) Abacavir - Drug Example

Figure 3.3: NICE Website - Screenshots[6]

(3.1). In order to find useful data in all doses I used regular expression to extract relevant information.

Most of the doses have the dosage and the unit near the beginning of the string, therefore the dosage will be extracted first. This always begins with an integer, while the following word is always the unit, hence the dosage in (3.1) is "600" and the unit "mg" Sometimes the dose might include "-" or "/", for instance in Figure A.5 where "300-900" would be the dosage. In such a case and similar cases including "/" I randomly selected one of the values which will be used as the dosage.

A similar special case can occur for units, for example "mg/kg" which is a possible unit in the data set but which would never be found on a prescription, as doctors always would calculate the dosage knowing the weight of the patient. To make my synthesised data as valid as possible I used a weight distribution to generate adult weights. This distribution includes a total observation number of 646 people and found the following statistics for adult weight(kg): minimum 53.72kg, mean 82.35kg, standard deviation 1.16kg and a maximum weight of 148.71kg. [26]

The remaining string will then be searched for any possible frequency including:

- "x times a day"
- "every x hours/minutes/.."
- "once/twice daily/weekly (in $(x - y)$ divided doses)"
- "once a month/in divided doses/ for x dose(s)"

where x and y are placeholders for integer values. After finding one of these regular expressions either the expression itself or one of its medical synonyms/abbreviations will be used. In (3.1) the frequency would be "daily in 1-2 divided doses" which is the remaining part of the dose and therefore the last step before the classification for this specific data

point would be completed. As an example, if having "4 times a day", that would lead to a random selection of either "QDS" or "q.d.s" in the prescription. Similarly other regular expressions as "once daily"/"every day"/"daily" would lead to "OD"/"o.d."/"once daily" or "ever day" in the prescription, which is similar for any other frequency given in the list above.

Looking at the example in Figure A.5 the remaining text of the dose would be "as required; maximum 4 g per day". Hence, the next step is to determine if the string contains anything related to the starting time of a prescription. Thus could be: "in the morning", "before food", "after food", "at night", "at bedtime", "in the evening", "immediately", "when required", "as required",.... Similar to the frequencies either the found words or the corresponding abbreviations will be used, as "ac"/"AC"/"a.c." for "before food" or "prn"/"p.r.n"/"PRN" for "as/if/when required", for the prescription. Whenever a doctor is writing a prescription, I assume that a phrase from the 'Starting' class will either have "start" or "starting" up front, therefore a random string of these two will be added to every instance of class 'Starting Time'.

Other important classes are the 'Maximum' and the 'Duration' class. To extract any instances of the first class, I am looking for a word group starting with "max" and will append all words to the class until facing a symbol such as: ",","."," ;",")","."),.... In the prescription data set, all instances of class 'Maximum' start with "Max". The opposite, rarely used class is the 'Minimum' class which will only be used if anything as 'to be given over x hour(s)/minute(s)' is found in the dose, any 'Minimum' class instance would start with "Min" in the prescription. The last possible class found is the 'Duration' class, which includes regular expressions such as "for x-y day(s)/dose(s)/week(s)/month(s)/year(s)" with integers x and y.

3.1.3 Synthesising data points

After getting all regular expressions, I have 281 individual drug prescriptions which will be split into 10 disjoint folds in order to be able to use 10-fold cross validation properly (as described in section 2.3). From each of these 10 folds, I generated 1000 samples using sampling with replacement, added noise to each sample and used a random order of classes for each sample. This is subsequently explained in sections 3.1.3 and 6.1.

Using 10-fold cross validation, the resulting average of any performance indicator is likely somewhat of an underestimate for the true indicator when trained on all data and tested on new (unseen) data. But the estimate is reliable in most cases, especially with a sufficiently large labeled data set and if the train and test data set follow the same distribution. [21]

Adding noise to synthesised data

After sampling with replacement I will randomly add noise to any extracted information to prevent and simulate spelling mistakes. Reasons for any mistakes could include motor control of hands and fingers, distance between the keys, visual similarity of characters, position in a word, same character repetition and phonological similarity of characters and words. [27] I am using the following four descriptive types:

- Deletion
- Repetition
- Substitution
- Transposition

Throughout, these 4 types each will be applied on any entry of a string with a probability of 2.5%, with some exceptions: never delete a character or number which stands alone;

any number, lower or upper case character can only be substituted by the same type.

Random Order of classes

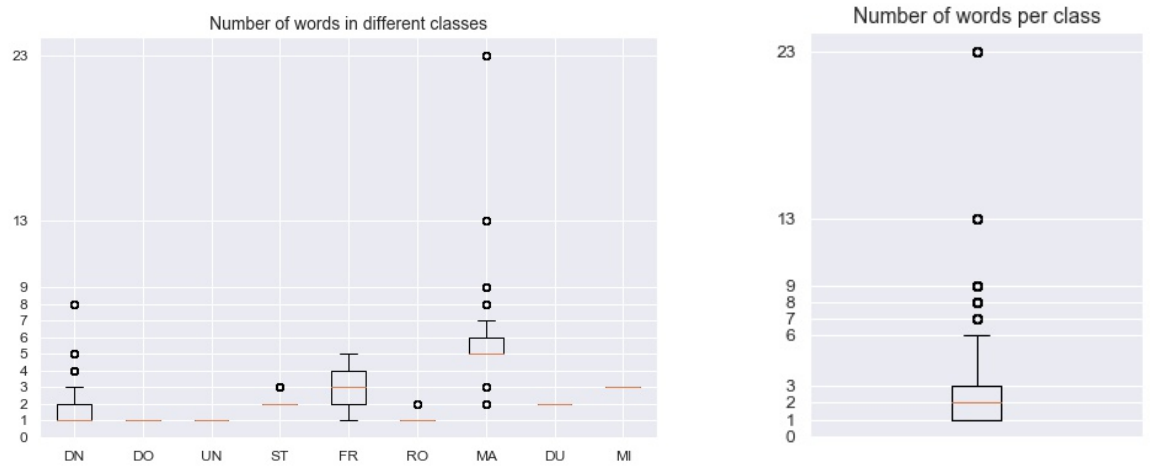
After the implementation of noise I will now randomly select class orders individually for every sample. In a later section (section 6) I will assume an underlying class order, as a single person's prescriptions would not be random but in a similar order every time due to human habits. I expect that with such an assumption the system will work even better.

Data Analysis

After synthesising 10,000 data points, Figure 3.4a shows that classes 'Dose' and 'Unit' always have one word, 'Duration' two words and 'Minimum' three words. Instances of class 'Starting Time' are 75% two words and the remaining quarter three words, and class 'Route' mostly has one word but sometimes also two. For classes including 'Drug Name', 'Frequency' and 'Maximum' data varies a bit more within the class. Especially within class 'Maximum' a hugely differing number of words can be found, with a minimum of 2 words, a median of five words and a maximum of 23 words. Overall Figure 3.4b shows that the number of words in a class has a mean of two words.

3.1.4 Drug Interactions

BNF's drug interaction list, `https://bnf.nice.org.uk/interaction/`, will be implemented to assist clinicians when prescribing a drug. If several drugs are given at the same time they can exert their effects independently or they can interact with each other. Many interactions are beneficial or harmless, although adverse drug interactions require special consideration. Therefore BNF implemented a relative importance of interactions as provided in the following list:



(a) Separated Classes

(b) All Classes together

Figure 3.4: Boxplots - synthesised data

Levels of severity:

- **Severe** May result in life-threatening event or permanent detrimental. effect
- **Moderate** Could cause considerable distress of partially incapacitate a patient, but are unlikely to be life-threatening or result in long-term effects.
- **Mild** Unlikely to cause concern to the majority of patients.
- **Unknown** Insufficient evidence to hazard a guess regarding the outcome.

Levels of evidence:

- **Study** Information based on formal study.
- **Anecdotal** Based on a single or a limited number of case reports.
- **Theoretical** Predicted interactions due to theoretical considerations.

Furthermore each interaction contributes effects which might occur, called an 'action message', and the actions to be taken, both contributions are based on manufacturer's advice or from relevant authorities.[6] For instance, in figures A.6a and A.6b you find BNF's interaction list of Abacavir and Aspirin, although Aspirins list is too long to be inserted, with a total number of 106 possible interactions.

For data extraction, I used regular expressions, more precisely I scraped each possible interaction for its level of severity and added those with severe and moderate level to the interaction lists. To demonstrate the interaction warning feature, I generated pseudo patients, with randomly selected medication lists between one and ten drugs. For simplicity reasons I have not checked the plausibility or possible interactions within these randomly selected medication lists. The underlying data set for random medication include all possible interactants of drugs starting with letter 'A'.

3.2 Tagging Models

Now, as all data has been generated, I tested several models including the first-order HMM Tagger, an Averaged Perceptron Tagger and the N-gram Tagger as explained in section 2.2. For the N-gram tagger backoffs using all smaller values N starting with $N-1$ will be included. In the worst case, the possible final backoff will be a default Tagger of class 'Drug Name' ('DN') as throughout the whole training data the most number of words have tag 'DN'. I also try to find the best N, in terms of computational time and overall success rate, and therefore will train and test the tagger with values of $N = 1, \ldots, 9$.

To be able to train and test properly I am using the already generated data of 10 disjoint folds and will use the 10-fold cross validation as in section 2.3 explained. After every model has been tested on all 10 folds the measurements will be averaged, only the overall confusion matrix will be the sum of all confusion matrices. After computing all

averaged measures the models can be compared and figured out which fits best. All of this will be done in Chapter 4.

3.3 Suggestion List

The system should not only be able to classify the final prescription correctly (so as to populate the fixed background prescription form), but should also suggest prescriptions as soon as the input has a length of at least 3 characters. These suggestions should not be random but rather those with the highest probability, created using all known data. In advance the system should draw the users attention to possible input spelling mistakes. To be able to do so, some further methods have to be initialised.

3.3.1 Next Word Class probabilities

The initial goal is to predict class t_n of word w_n of an input string. Therefore all estimates $\hat{t}_1, \ldots, \hat{t}_{n-1}$ of all previous words have to be estimated first using a POS Tagger. After that the generalised MLE N-gram parameter estimation is used for N=1,...,5:

$$\begin{aligned}
Unigram: &\quad P(t_n) = \frac{C(< s >, t_n)}{C(< s >)} \\
Bigram: &\quad P(t_n|\hat{t}_{n-1}) = \frac{C(\hat{t}_{n-1}, t_n)}{C(\hat{t}_{n-1})} \\
Trigram: &\quad P(t_n|\hat{t}_{n-2}, \hat{t}_{n-1}) = \frac{C(\hat{t}_{n-2}, \hat{t}_{n-1}, t_n)}{C(\hat{t}_{n-2}, \hat{t}_{n-1})} \\
4-gram: &\quad P(t_n|\hat{t}_{n-3}, \ldots, \hat{t}_{n-1}) = \frac{C(\hat{t}_{n-3}, \ldots, \hat{t}_{n-1}, t_n)}{C(\hat{t}_{n-3}, \ldots, \hat{t}_{n-1})} \\
5-gram: &\quad P(t_n|\hat{t}_{n-4}, \ldots, \hat{t}_{n-1}) = \frac{C(\hat{t}_{n-4}, \ldots, \hat{t}_{n-1}, t_n)}{C(\hat{t}_{n-4}, \ldots, \hat{t}_{n-1})}
\end{aligned} \tag{3.2}$$

Which will then be for linear interpolation (see equation 2.14) to estimate the 5-gram probability $P(t_n|\hat{t}_{n-4}, \ldots, \hat{t}_{n-1})$. But instead of finding the optimal set of λs with the EM

algorithm for example, it will be assumed that all λs have the same value and therefore it is rather an average than an interpolation:

$$\hat{P}(t_n|\hat{t}_{n-4},\ldots,\hat{t}_{n-1}) = \sum_{i=1}^{4} \frac{1}{5} P(wt_n|\hat{t}_{n-i},\ldots,\hat{t}_{n-1}) + \frac{1}{5} P(t_n) \tag{3.3}$$

I also want to force the model to generate only valid prescriptions, meaning that for any word group with the same predicted tag for the first and last word, all words within the group have to have the exact same predicted tag, therefore a constraint has to be added to the equation. The model can only guarantee valid prescriptions if all predicted classes of previous words cannot be the class of any of the following words, hence their probabilities in equation 3.3 will be set to 0 from the beginning, excluding the predicted class of the last word $\hat{t}_{n-1}$. If any classes' probability was changed to 0 because of that, probabilities have to be recalculated for a valid probability distribution.

3.3.2 Autocomplete a word itself

To be able to autocomplete a word itself, standard probabilities have been used including all words of the data set, which is not immensely large and therefore any more sophisticated probability models are not required. If a word w is formed by characters $c_1,\ldots,c_n$ any possible truncation of the character sequence, $(c_1),(c_1,c_2),(c_1,c_2,c_3),\ldots$ was added to a dictionary, where each truncation is pointing to the word w itself. As soon as different words share some of the starting letters, relative frequencies were implemented, which were used as probabilities in the autocompletion model. For instance, if the current word would be "asp" within the data autocompletion options would be "aspirin" and "asparaginase" with a probability of 92% and 8% respectively.

3.3.3 Generate words to suggest proper prescriptions

For a proper suggestion the current string has to be tagged first, and the created sequence will be used to get the next word class probability as described in section 3.3.1. Then the MLE 5-gram parameter estimation model will be considered as in equation 2.13, but now to generate a possible following word and check if its corresponding next word class probability is nonzero. If it is zero the process will be repeated and tried to generate other words which hopefully then are probable next words. After finding a word the whole process will be repeated for the new extended string until a proper prescription string was found. This repetition will be interrupted as soon as it cannot find any new valid words, which can be added to the current word sequence.

3.3.4 Suggest prescriptions

The suggestion list will then be built up in the following way:

1. Check the input for possible spelling mistakes and suggest its correction,
2. Autocomplete last input word in combination with its class prediction,
3. Form a list of maximum 6 prescription suggestions.

First, the present input, $w_1, \ldots, w_n$, will be searched for any spelling mistakes. Such spelling mistakes are located using a dictionary, including any misspelled words within the training data pointing to its correct word counterpart. For all other suggestions it will use the input as the beginning of the string, ignoring the spelling mistake, as the algorithm might find a misspelling even though it is just a new word the algorithm has not seen so far. Subsequently the next word probabilities will be computed, as described in section 3.3.1, of the sequence $w_1, \ldots, w_{n-1}$, assuming that the last word has not been finished yet. To finish the word w_n it will be autocompleted ,as described in section 3.3.2, and checked

whether its predicted tag has nonzero probability to be the next word. The list will then be filled up generating suggestions, as explained in section 3.3.3, up to a maximum of six list entries in total.

Chapter 4

Evaluation

4.1 Tagger selection

The best way to evaluate language model performance is called extrinsic evaluation, which embeds it in an application and measures its improvement. Thus, all considered models can be compared by running every model and checking which provides the most accurate output. [15]

4.1.1 *N*-gram selection

First, I will evaluate which N-gram model fits best, which will then considered for further evaluation. Therefore the Unigram, Bigram, Trigram, 4-gram,..., 9-gram model was computed using the 10-fold cross validation to get the average OSR and the overall computational time. Results are plotted in Figure 4.1, with an exponentially increasing computational time starting with 22 seconds for the 1-gram model up to 338 seconds for the 9-gram model. Looking at the OSR a jump from the Unigram model with a value of 0.8378 to 0.8652 of the Bigram model can be seen. After which a minor decrease follows down to 0.8645 for the Trigram model and the value gets back from the 4-gram model

onwards and fluctuates between 0.8654 and 0.86589. Consequently the Bigram model will be considered for further in-depth analysis, due to its combination of low computational time and a reasonably high OSR value.

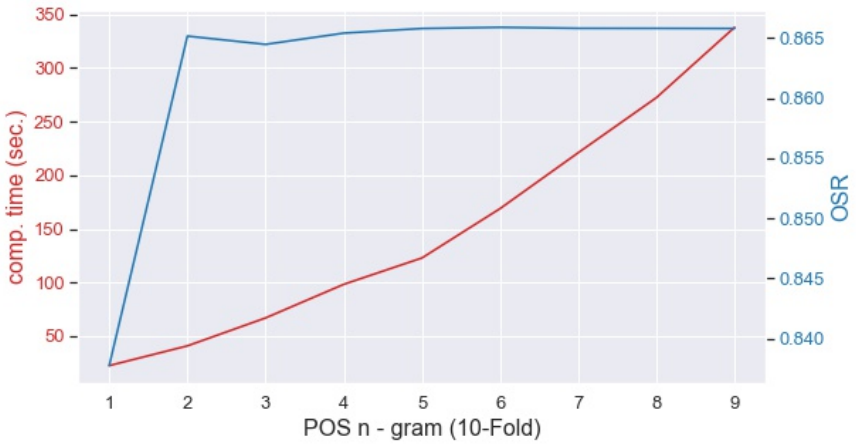


Figure 4.1: N-gram analysis

4.1.2 Performance measurements of all three models

To be able to evaluate and measure performances of all models the confusion matrix, OSR and all comparison metrics as explained in section 2.4 will be considered. The 10-fold cross validation was used to ensure a reasonable comparison between model measurements by averaging over results over all 10 folds. Only confusion matrices will be summed up instead, which is the common way of doing so. In table 4.1 all measurement values are listed and in figures 4.2, 4.3 and 4.4 all test results' confusion matrices of the respective models are shown. To get a better feeling for the results I used a colormap plot, where the background color of each cell (i, j) is dependent on the proportion $p_{i,j}$. If proportion $p_{i,j}$ is 1 the background color would be white, if it is 0 black, if 0.5 red and all other values would have a mixture of colors dependent on the corresponding proportion. If all elements of the testing would have been classified correct, all off diagonal elements would be black

with proportion 0 and the diagonal being white with proportions 1.

Figure 4.2 shows the confusion matrix of the first order HMM Tagger model. At first glance the brightness of the most left column sticks out, which is only positive for the first class ('DN'), but shows a very bad prediction for all other classes which mostly are predicted wrong. More precisely, the proportion of any classes word being predicted to be from class 'DN' is higher than 0.48, and is highest for words from class 'MI' with a proportion of 0.67, which additionally is the class with the lowest true positive rate and an OSR of only 0.14. Specifically, looking at class 'DN' its false positive value going through the roof, with $n_{FP}^{\text{'DN'}} = 43,433$ leading to a low $TNR_{\text{'DN'}}$ and $PPV_{\text{'DN'}}$. Hence, all other classes have a high false negative value.

In contrast, the confusion matrix of the Bigram Tagger model (Figure 4.3) shows a better classification. The diagonal proportions are mostly above 0.8 and are even above 0.9 for classes 'DN', 'ST' and 'RO'. Only classes 'MA', 'MI' and 'DU' do not have such high proportions in the diagonal and similarly have comparingly low PPVs. For words actually coming from 'MA' class the model sometimes predicts 'DO' instead with a proportion of 12%. Whereas 11% of actual 'DU' class words are classified as 'DO', 10% as 'DU' and 17% as 'FR'. The worst classification has class 'MI', with the lowest OSR of 0.4 and 29% wrongly classified as 'FR', 14% as 'DU' and 9% as 'DO', although this class is facing by far the lowest population throughout all classes with an overall 'MI' test words of 753.

The confusion matrix of the Averaged Perceptron model (Figure 4.4) looks even better than the one with the Bigram model (Figure 4.3). Similar to Figure 4.3 the first six classes have a high OSR, only this time significantly higher as five proportions are above 0.97 with one actually hitting 1.00 and one with 0.95. Class 'DU' has an OSR value of 0.92, therefor seven OSR values are above 0.9. Only leaving classes 'MA' and 'MI' below 0.9, but not that far with 0.83 and 0.79 respectively, and both classified considerably better compared with the previous model.

Overall, looking at the averaged comparison metrics in table 4.1 similar conclusions can be made that the Averaged Perceptron model is predicting best. Overall the Perceptron algorithm classified 7,827/10,000 (78.27%) test prescriptions completely correct.

Model	OSR	AAcc	ER	P	R	F-Score
First-Order HMM	0.493	0.887	0.113	0.763	0.412	0.533
Bigram	0.865	0.970	0.030	0.775	0.746	0.760
Averaged Perceptron	0.958	0.991	0.009	0.859	0.854	0.857

Table 4.1: Average of all 10-Fold Measurements

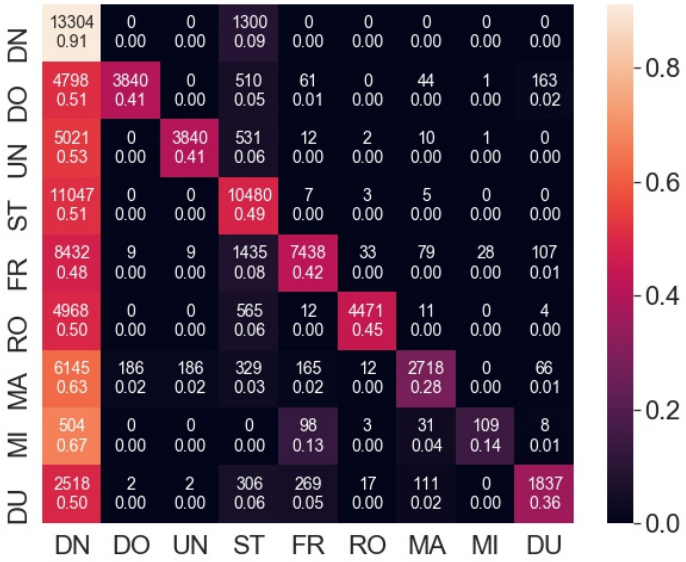


Figure 4.2: Colormap - Confusion Matrix - First-Order HMM

The main goal of the project is to improve the currently used interface by using a combination of two separate functions in the background of a single graphical user interface (GUI). When users want to prescribe a drug, they would start typing into a single text box. Meanwhile, in the background, the first function, which will be called 'prediction' function, autocompletes the input to a list of possible prescriptions, using Natural Language

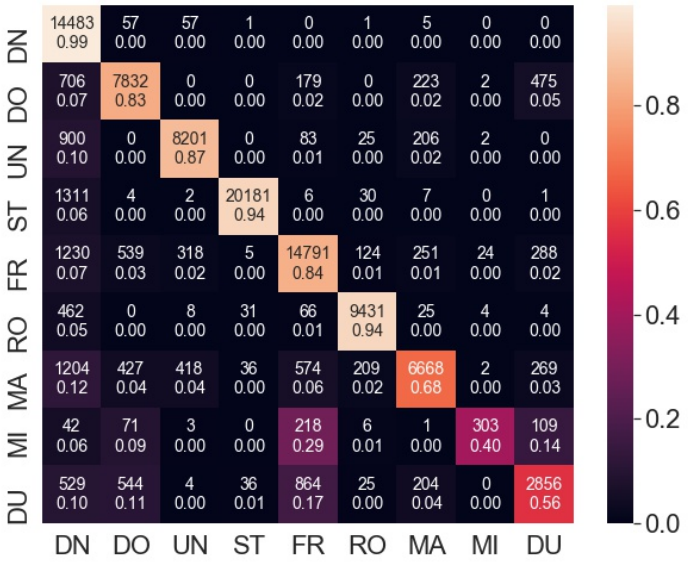


Figure 4.3: Colormap - Confusion Matrix - Bigram

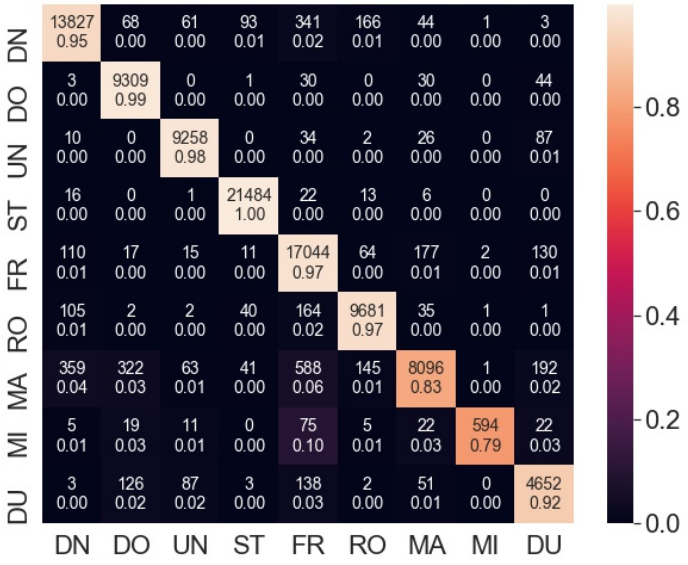


Figure 4.4: Colormap - Confusion Matrix - Averaged Perceptron - Random Data

Processing (NLP). To be able to provide such a suggestion list a trained Tagger model is used to classify the input portion, and predict the next word class and its corresponding word. Where the training data of the Tagger model is a set of web scraped and synthesised prescriptions from BNF's website. After the user has selected a suggestion or finished the free text input, it will be split up by the Tagger model into its corresponding class fields in the GUI. For instance, the Tagger model should be able to classify and split up a prescription shown in (1.1) into its corresponding classes, where classes can be found beneath each word. However, in actual deployment the 'prediction' function would be trained on prescribers' historical prescription records and thus be tuned to each individual's unique prescription and writing style. The second function, called 'warning' function, uses the BNF database of known drug interactions to provide a user with warnings about any possible adverse reaction for a specific patient, knowing a patient's current medication list. If the level of severity of the interaction is sufficiently high, the corresponding drug will be highlighted in the suggestion list and a warning pops up showing relevant information, including the drugs the interaction risk is coming from and its level of severity.

Chapter 5

Improving Drug Prescription Processing

How a prescription will be processed in a UI is one of the most important components in electronic prescriptions, and was the initial motivation to write the thesis. Prescribing a drug is expected to require sufficiently less training and to be a straight forward tool to use for any medical staff. Therefore it has to be simple and efficient, with its main requirement to decline human errors in the whole prescription process.

To demonstrate the new prescription process and the power of natural language processing behind it, I developed a GUI which incorporates all parts of the 'prediction' and 'warning' function, as defined in section 1.2. The model architecture behind of the user interface was already shown in chapter 3 Figure 3.1. The graphical user interface is shown in figures 5.1, 3.3, 5.5 and 5.4, with the first two, showing the part of the prescription process related to the 'prediction' function and the other two qre related to the 'warning' function.

As shown in Figure 5.1, the GUI has one input field for all prescription related information, called 'Prescription'. As soon as three characters are entered the 'prediction'

function is trying to generate a suggestion list, using the Averaged Perceptron Tagger model within the whole generating process, as explained in section 3.3. Figure 5.2 shows, that the suggestion list was realised as a drop-down menu in the user interface, with a maximum length of six entries and no minimum length, as it might be the case that the 'prediction' function is not able to generate any suggestions. As the user alters the input, the suggestion list will be adapted. Whenever the algorithm finds a possible misspelling in the input, it will add a correction suggestion to the top of the suggestion list, see Figure 5.3. Additionally, Figure 5.3 shows how the suggestion process is handling possible misspelled input. In Figure 5.3 the entered input prescription starts with "aspiirin", the process will now check the misspelling dictionary, explained in section 3.3.4, finding word "aspirin" as its possible correction. Only the first suggestion will then include the potential correction "aspirin" along with the actual input string. For generating suggestions, the actual input will be considered to generate them, as the user might has written all words on purpose. To select any suggestion, the user has to double-click the respective suggestion, which will then be inserted the 'Prescription' input field.

After the user has selected or typed in the final prescription, the second part of the 'prescription' function will be started by pressing the 'Split Up' button. The final prescription input will then be classified using the Averaged Perceptron Tagger and depending on its classification output the words will be inserted into the corresponding class text field, as can be seen in Figure 5.1. The prescription input in Figure 5.1 is the same as in Example 1.1 of Chapter 1 and was classified as expected. The classification of the two other examples, 1.2 and 1.3, can be found in the appendix, Figure A.8a and A.8b. If the user wants to delete all inputs, the button 'Clear All' can be pressed and delete all the data within the prescription process. Whereas, the consequences of pressing button 'New Patient' and ticking the checkbox 'Interactions' will be explained in the next section.

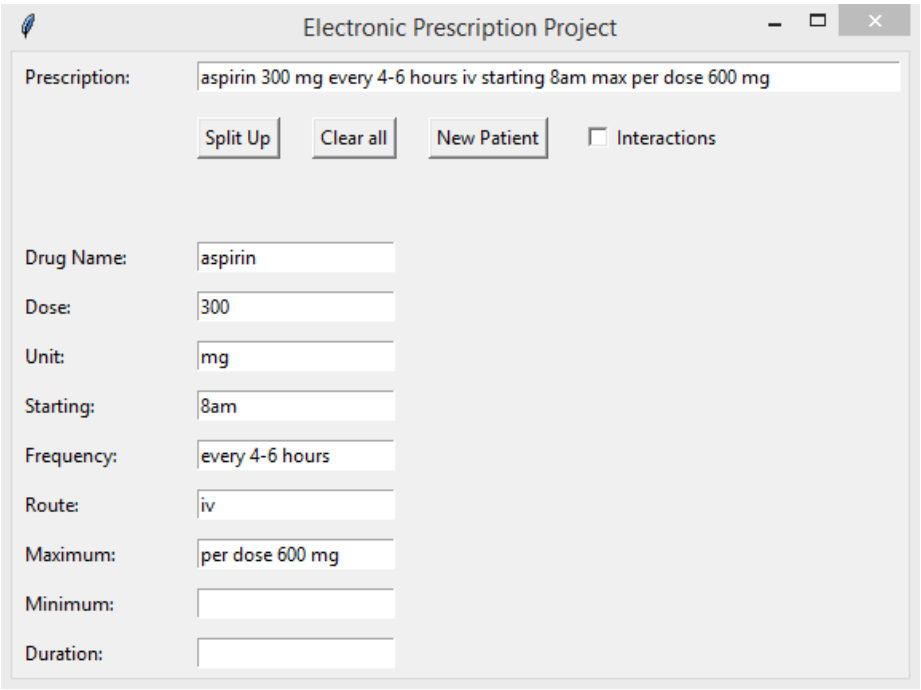


Figure 5.1: Classification of Example 1.1

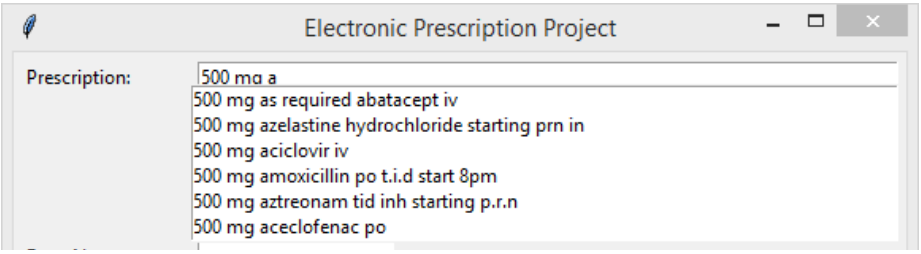


Figure 5.2: Suggestion List Example

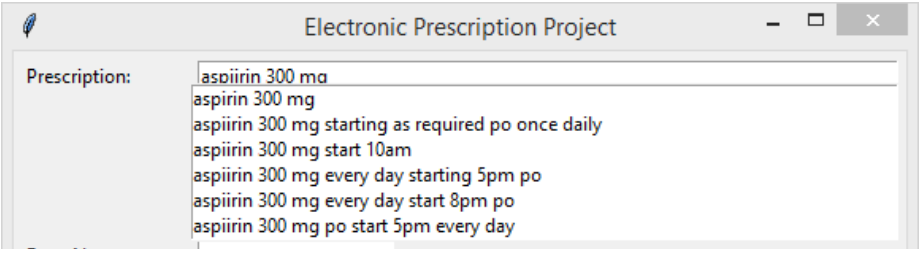


Figure 5.3: Suggestion List Example Spelling Mistake

5.1 Interactions

The 'warning' function of the prescription process can be activated by ticking the 'Interaction' checkbox, which will assist clinicians when prescribing a drug. At first the 'warning' function will generate a random medication list of maximum size 10, as defined in section 3.1.4, simulating an actual deployment where the prescription process would have a patient's specific medication. This random selection can be resampled clicking at button 'New Patient' in the user interface. For instance, in figure 5.4 the patients current medication would include Promethazine, Aripiprazole, Cisplatin, ..., Darunavir and Naproxen. Thereby, if the user is again inputting at least three characters to the 'Prescription' field the 'prediction' function will also be activated. Now, both functions are activated and therefore any drug suggestion having at least a moderate risk of interaction with any drug of the current medication list will be highlighted depending on the level of severity. If the level of severity is moderate a suggestions background color will be yellow and if it is severe it will be highlighted with color red. In Figure 5.5 the same current medication as in Figure 5.4 has been used including a 'Prescription' input of "300 mg a", which leads to the generation of a suggestion list including three suggestions without a risk of interaction, one with moderate and two with severe interaction risk. The one with moderate severity is because of the interaction risk of drugs Alitretinoin and Darunavir, details can be found in Figure A.9. The remaining two prescriptions, highlighted with background color red, have both a severe level of severity due to the theoretical interactions between Aspirin and Nicorandil, see Figure 5.5. However, a user might still want to prescribe a drug with interaction risk, following that the prescription process pops up a warning in the UI containing BNF's interaction statement in detail, highlighted with background color red or yellow respectively.

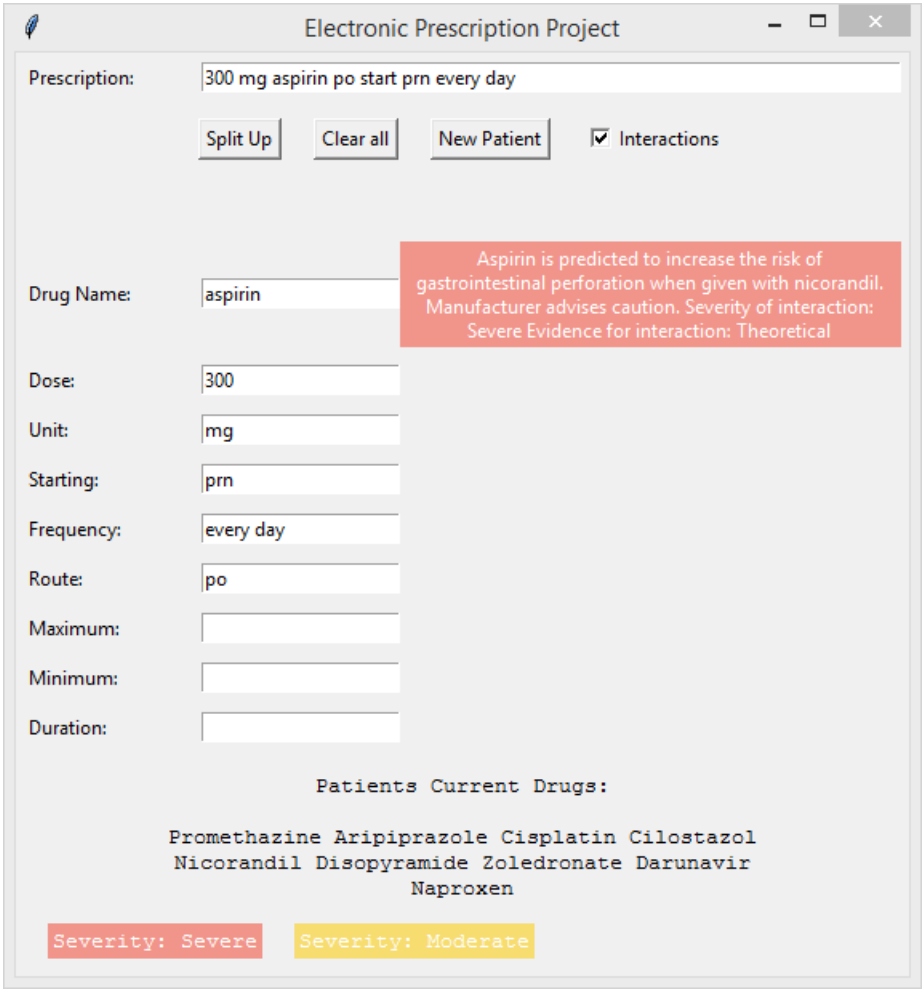


Figure 5.4: Interaction example

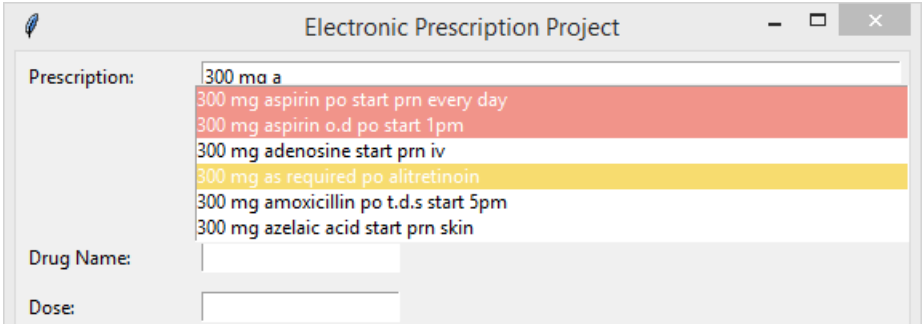


Figure 5.5: Interaction Suggestion List

5.2 Special Examples

This section is about two theoretical examples which I input to demonstrate the flexibility of the implemented prescription process. In Figure 5.6 the prescription input is the random string "asdfghjklo 123 m". Even though the Tagger model predicts the first word to be from class 'DN' and "123" from 'DO' and the prescription process finds reasonable suggestions, even for such a random input. Additionally, Figure 5.6 shows possible mistakes within the suggestion list, the 'prediction' function sometimes suggests inappropriate or unreasonably truncated prescriptions. For instance, in Figure 5.6 the third suggestion includes "per course" which does not make sense as such. Suggestions four and five are both truncated at the end, as "max duration of treatment" and "max" as such, would not make any sense in a prescription. Even though, truncated suggestions are less problematic as the user could still select them and add words to create reasonable prescriptions. These three suggestions in Figure 5.6 are not reasonable prescriptions, but they are 'valid'. Any prescription is said to be 'valid' if for any group of words of the prescription, having the same predicted tag for the first and last word, all words within the group have the same predicted tag. This is not always the case, as can be seen in Figure 5.7. In this example the prescription input includes the word "aspirin" twice and is predicted by the Tagger model to have the following predicted classes: ['Drug Name']['Dose']['Unit']['Drug Name']. Therefore, it is an 'invalid' prescription, as if you take the complete input string as a word group, which has the same predicted class for the first and last word but all others in between have a different predicted class. Another situation which would also possibly lead to an 'invalid' prescription, would be the case of a long, unknown and rather complicated prescriptions, which might not be actually 'invalid' as such but the model fails to classify it correctly. In both of these situations the prescription process will alert the user with a warning message, saying "Warning: Invalid", in the user interface.

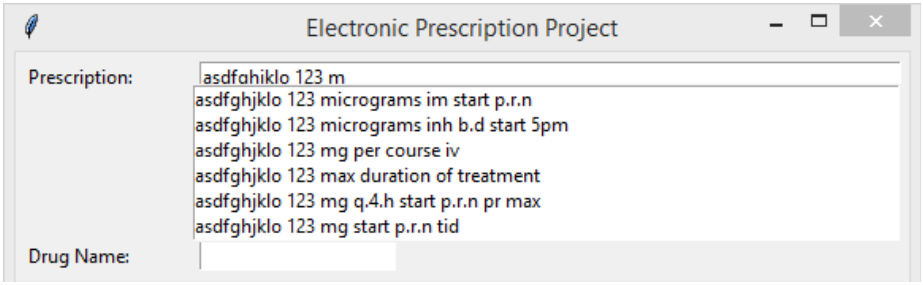


Figure 5.6: Random input

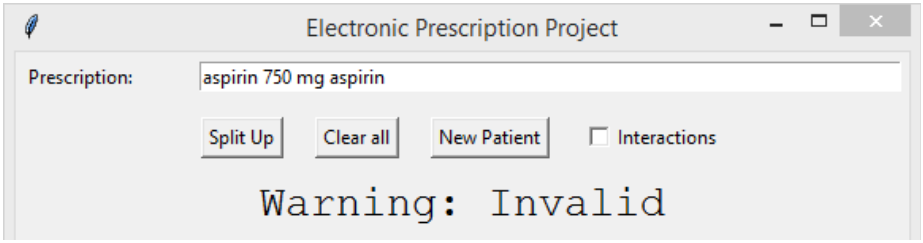


Figure 5.7: Invalid input

Chapter 6

Experimental Results

6.1 Logical Prescription

As humans are creatures of habit and rarely change their routines, the general assumption made in this section, is that every specific user has an underlying class order when prescribing a drug, meaning a user sticks to a specific class order. Therefore the random class order in the data synthesising process, in section , will now be ignored and the following order will be used for every prescription instead:

‘Drug Name’, ‘Dose’, ‘Unit’, ‘Starting Time’, ‘Frequency’,
‘Route’, ‘Maximum’, ‘Minimum’, ‘Duration’

Using this assumption the evaluation has to be recalculated for the new training data. I expect a better performance as the underlying model is now fixed, hence all models should work better.

6.2 Logical Prescription - Evaluation

The evaluation process is the same as in Chapter 4, therefore all three models will be evaluated with the newly ordered data by using all performance metrics with 10-fold cross validation.

The evaluation results can be found in table 6.1 and Figure 6.1. The first-order HMM model is worse now and only reaches an OSR value of slightly above 0.203, which is significantly lower to a value of 0.493, having randomly ordered data. Whereas both other models meet the expectations of achieving better results with an increase from 0.865 to 0.889 of the OSR value for the Bigram model and a similar increase for the Averaged Perceptron model from 0.958 to 0.979, this increase is proportionally similar looking at the F-Score of these two models.

For detailed reasoning, the confusion matrices have to be compared, see Figure 6.1 and 4.4. Class 'Starting Time' was now able to reach a perfect OSR value of 1.0, as were classes 'Dose' and 'Unit'. Although, the OSR value of class 'Duration' did not improve that much, it still reaches a reasonable value of 0.93, class 'Maximum' now gets classified weigh better with 0.91 compared to 0.83 before. The only class where the OSR value decreased is class 'Minimum', from 0.79 to 0.52, although the data set for this class is very small. Even though, the proportion of words of class 'Minimum' being classified as 'Maximum' jumped from 0.03% to 0.22%, which is the same ratio as for wrongly predicting 'Maximum' words to be in class 'Duration'. The overall ratio of prescriptions having at least one word misclassified is 12.96% (1,296/10,000), without invalid prescriptions.

Therefore using the new trained Averaged Perceptron Tagger within the prescription process, the 'prediction' function would perform slightly better due a higher OSR. But suggestions tend to be worse as the system is now much more sensitive to changes of the input. This results do not meet my expectations, although the tailored sentence order is

increasing the performance of the classifier, but the generated suggestion lists are of poor quality.

Model	OSR	AAcc	ER	P	R	F-Score
First-Order HMM	0.203	0.823	0.177	0.562	0.156	0.231
Bigram	0.889	0.975	0.025	0.812	0.772	0.791
Averaged Perceptron	0.979	0.995	0.005	0.878	0.867	0.872

Table 6.1: Average of all 10-Fold Measurements (Habit Data)

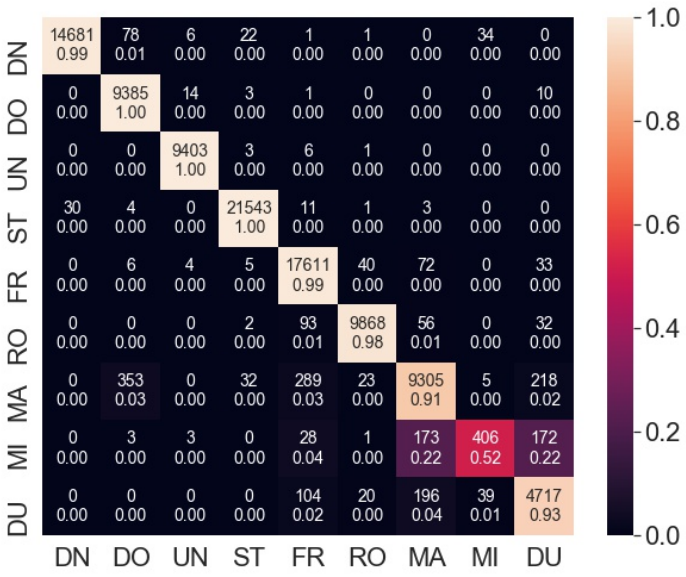


Figure 6.1: Colormap - Confusion Matrix - Averaged Perceptron - Habit Data

Chapter 7

Conclusions and further work

I will now analyse the outcomes of my thesis, especially if they meet the initial list of requirements, defined in section 1.2, and give a short outlook for possible further work.

Firstly, the data set I used within my thesis was synthesised, to prevent confidentiality issues of real data, by web scraping BNF's website. I also used regular expressions to identify prescription related data to generate a reasonable prescription data set, which was a quite challenging part, as I worked out all regular expressions manually. This data set was used in the 'prediction' function of the prescription process. Whereas all data for the 'warning' function was also web scraped BNF's interaction database. Incorporating all example screenshot of using the test user interface in section 5, the synthesising process can said to be generating reasonable prescriptions. Although, it has to be said that prescriptions have underlying assumptions, as in section 3.1.1 explained. Another shortcoming of this way of synthesising data, is finding all regular expression manually, especially for bigger data sets it can very quickly become a demanding process to do so.

For the second requirement in the list, I compared using a first-order HMM Tagger model, different N-gram Tagger models and the Averaged Perceptron Tagger model in Chapter 4. Where I evaluated all models by computing different metrics and finding the

most accurate one by comparing these measurements, which lead to the Averaged Perceptron Tagger model as the best accurate one. Which has an OSR value of 0.958, precision, recall and F-Score all between 0.85 and 0.86 and therefore is a reasonably accurate classification model.

The following requirement, is part of the 'prediction' function in my prescription process and was implemented in Chapter 5, sample screenshots of suggestion lists were shown in different figures. The prescription process is able to suggest a varied list of prescriptions even for a very short prescription input. The implementation of the suggestion list was a challenging, as the process of selecting new suggestions should be fast but also should include valid and reasonable prescriptions. But the suggestion list definitely has its difficulties, as sometimes suggestions make no sense, as parts of classes are missing or the last class is truncated for example, and for a longer input, processing the suggestion list can get very time consuming.

The next requirement in the list was also implemented in 5 and belongs to the 'warning' function. Again if looking at respective figures in the thesis it can be said that this requirement was reasonably addressed within my process, by either highlighting drugs with an interaction risk in the suggestion list and also showing more details about the interaction problems when selecting a riskier drug.

For the final requirement, which was also the motivation of writing the thesis, I can only consider my subjective perspective, which is comparing the old and my prescription process I would definitely say my improvements simplify the process for end users, but to find a proper feedback for this requirement it would be better to let some clinicians test the system and compare it to the old system.

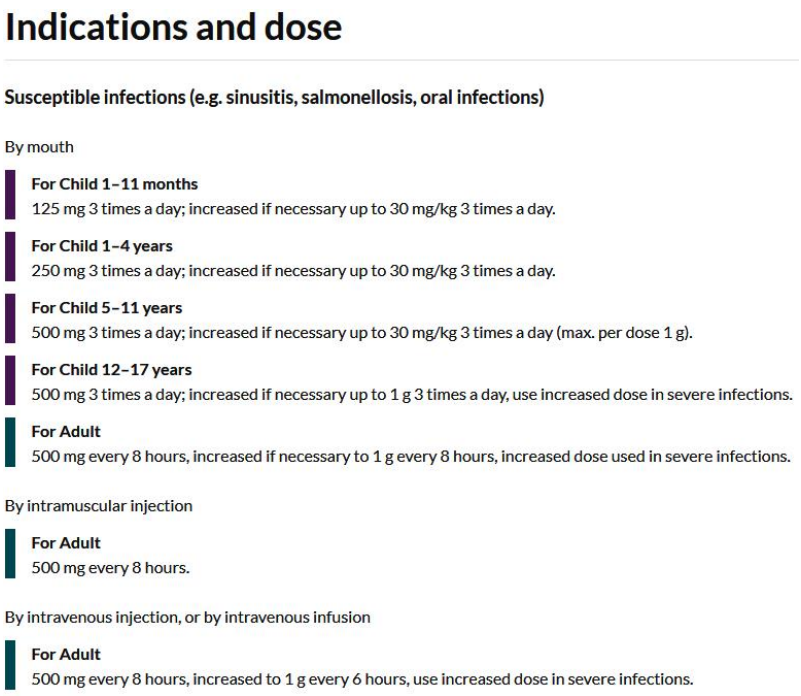
7.1 Further Work

In the following I will point out some ideas of possible further work in the topic of my thesis. I will do so in a similar order to the requirements. To be able to get a wider ranged suggestion list, a more complex data set would be needed by web scraping additional data sources, which would lead to a very time consuming process when trying to find regular expressions manually, therefore instead NLP methods could be used to extract important information as explained in [11]. Classification models could probably also be improved or at least some others could be evaluated, for example a more advanced HMM model or a LSTM. Whereas the suggestion list could be definitely improved by generating suggestions faster, especially for long input strings the current system can lag. A speedup could be achieved by analysing users' input behaviours and only generating a list when they are really expecting to get on. The correction of misspellings could be improved by using an NLP model instead of rarely changing database, an example can be found in [13]. Another improvement could be done by enhancing the interaction list, not only using known drug interactions but also using patients' diseases (as noted in their charts) to prevent interactions between specific diseases and drugs. Finally, the GUI could include the possibility to alter classified text and then submit it, which would lead to the possible incorporation of currently prescribed drugs into the training data of the Tagger model, which would then be retrained after several new training data points. This model could then be analysed how fast it can adapt to new habits of a user. The last idea I would have for possible further work, would be letting clinicians test the system and incorporate their feedback.

Appendix A

Additional Screenshots

A.1 BNF Screenshots



Indications and dose

Susceptible infections (e.g. sinusitis, salmonellosis, oral infections)

By mouth

For Child 1–11 months
125 mg 3 times a day; increased if necessary up to 30 mg/kg 3 times a day.

For Child 1–4 years
250 mg 3 times a day; increased if necessary up to 30 mg/kg 3 times a day.

For Child 5–11 years
500 mg 3 times a day; increased if necessary up to 30 mg/kg 3 times a day (max. per dose 1 g).

For Child 12–17 years
500 mg 3 times a day; increased if necessary up to 1 g 3 times a day, use increased dose in severe infections.

For Adult
500 mg every 8 hours, increased if necessary to 1 g every 8 hours, increased dose used in severe infections.

By intramuscular injection

For Adult
500 mg every 8 hours.

By intravenous injection, or by intravenous infusion

For Adult
500 mg every 8 hours, increased to 1 g every 6 hours, use increased dose in severe infections.

Figure A.1: Amoxicillin - Indication and Dose [6]

For Caverject®

Erectile dysfunction

By intracavernosal injection

For Adult
Initially 2.5 micrograms for 1 dose (first dose), followed by 5 micrograms for 1 dose (second dose), to be given if some response to first dose, alternatively 7.5 micrograms for 1 dose (second dose), to be given if no response to first dose, then increased in steps of 5–10 micrograms, to obtain a dose suitable for producing erection lasting not more than 1 hour; if no response to dose then next higher dose can be given within 1 hour, if there is a response the next dose should not be given for at least 24 hours; usual dose 5–20 micrograms (max. per dose 60 micrograms), maximum frequency of injection not more than 3 times per week with at least 24 hour interval between injections.

Erectile dysfunction associated with neurological dysfunction

By intracavernosal injection

For Adult
Initially 1.25 micrograms for 1 dose (first dose), then 2.5 micrograms for 1 dose (second dose), then 5 micrograms for 1 dose (third dose), increased in steps of 5–10 micrograms, to obtain a dose suitable for producing erection lasting not more than 1 hour; if no response to dose then next higher dose can be given within 1 hour, if there is a response the next dose should not be given for at least 24 hours; usual dose 5–20 micrograms (max. per dose 60 micrograms), maximum frequency of injection not more than 3 times per week with at least 24 hour interval between injections.

Figure A.2: Alprostadil - Indication and Dose [6]

Indications and dose

Cardiovascular disease (secondary prevention)

By mouth

For Adult
75 mg daily.

Management of unstable angina and non-ST-segment elevation myocardial infarction (NSTEMI), Management of ST-segment elevation myocardial infarction (STEMI)

By mouth

For Adult
300 mg, chewed or dispersed in water.

Suspected transient ischaemic attack

By mouth

For Adult
300 mg once daily until diagnosis established.

Figure A.3: Aspirin - Indication and Dose 1 [6]

Transient ischaemic attack (long-term treatment in combination with dipyridamole), Ischaemic stroke not associated with atrial fibrillation (in combination with dipyridamole if clopidogrel contraindicated or not tolerated), Ischaemic stroke not associated with atrial fibrillation (used alone if clopidogrel and dipyridamole contraindicated or not tolerated)

By mouth

For Adult
75 mg once daily.

Acute ischaemic stroke

By mouth

For Adult
300 mg once daily for 14 days, to be initiated 24 hours after thrombolysis or as soon as possible within 48 hours of symptom onset in patients not receiving thrombolysis.

Atrial fibrillation following a disabling ischaemic stroke (before being considered for anticoagulant treatment)

By mouth

For Adult
300 mg once daily for 14 days.

Following disabling ischaemic stroke in patients receiving anticoagulation for a prosthetic heart valve and who are at significant risk of haemorrhagic transformation

By mouth

For Adult
300 mg once daily, anticoagulant treatment stopped for 7 days and to be substituted with aspirin.

Figure A.4: Aspirin - Indication and Dose 2 [6]

Following coronary by-pass surgery

By mouth

For Adult
75–300 mg daily.

Mild to moderate pain,
Pyrexia

By mouth

For Adult
300–900 mg every 4–6 hours as required; maximum 4 g per day.

By rectum

For Adult
450–900 mg every 4 hours; maximum 3.6 g per day.

Acute migraine

By mouth

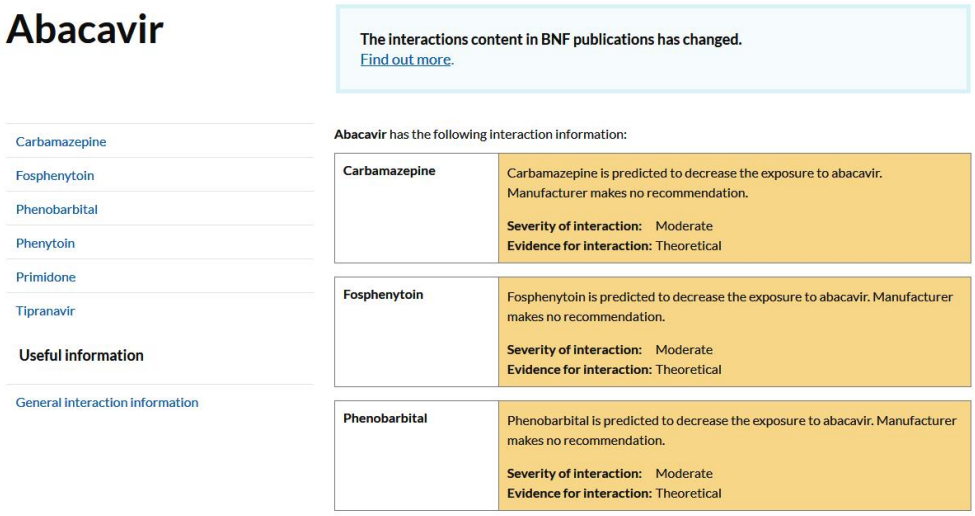
For Adult
900 mg for 1 dose, to be taken as soon as migraine symptoms develop.

Prevention of pre-eclampsia in women at moderate or high risk

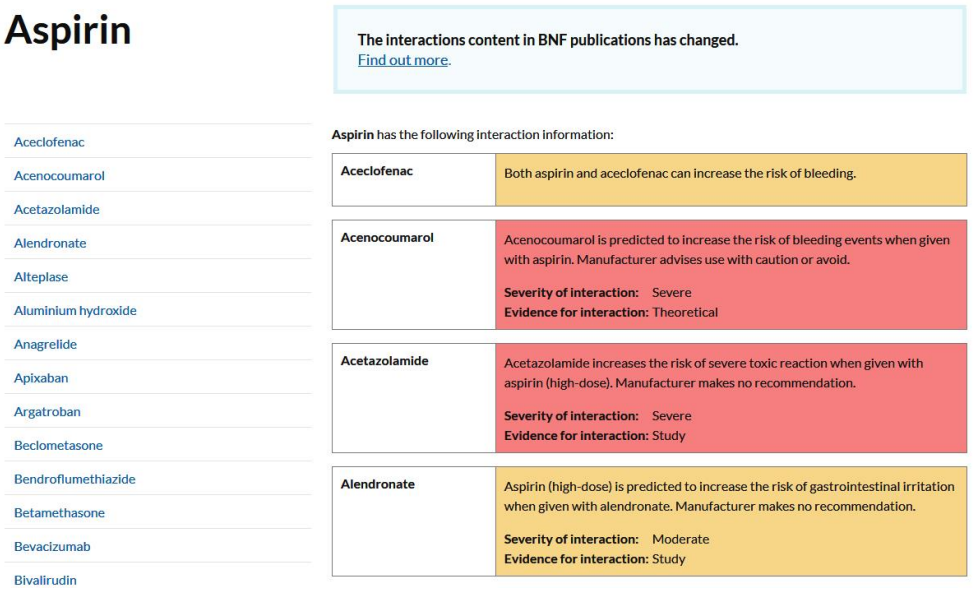
By mouth

For Adult
75–150 mg once daily from 12 weeks gestation until the birth of the baby.

Figure A.5: Aspirin - Indication and Dose 3 [6]



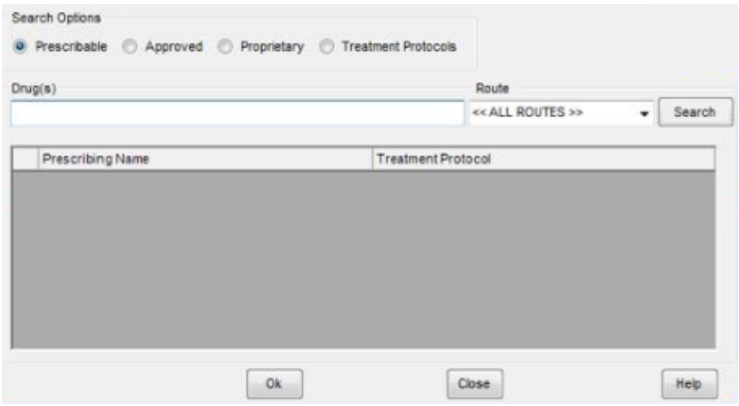
(a) Abacavir



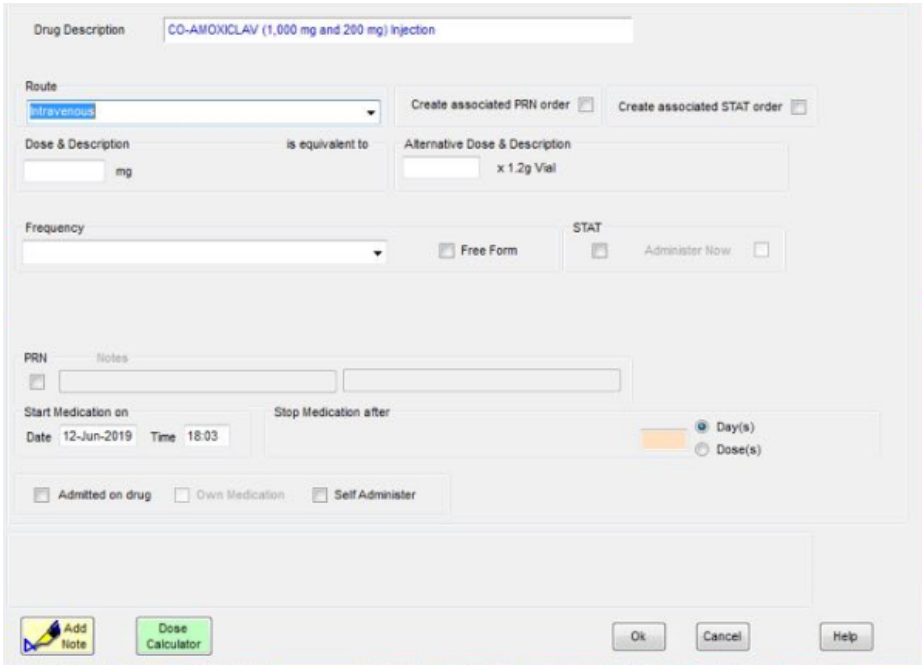
(b) Aspirin

Figure A.6: BNF - Drug Interactions [6]

A.2 Current GUI Screenshots



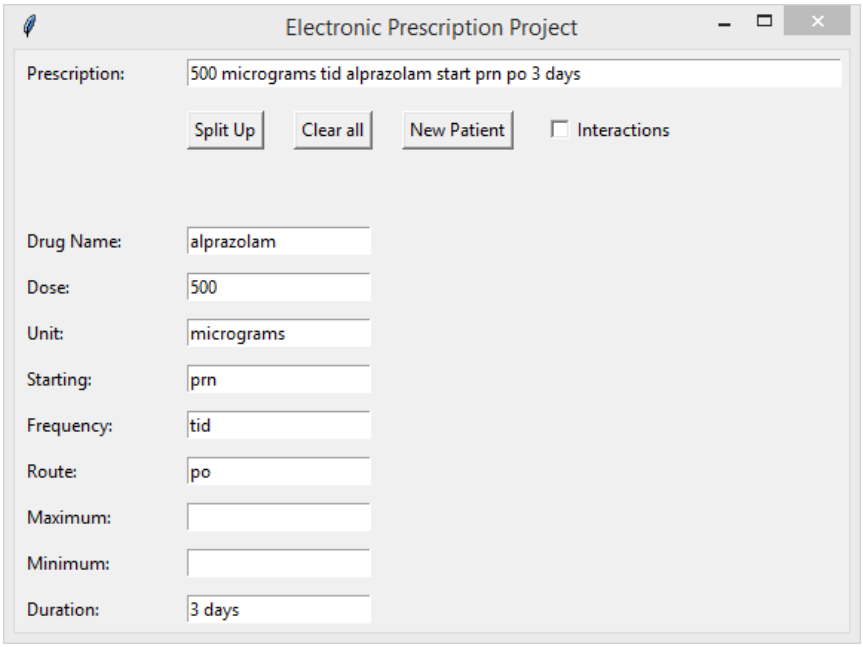
(a) Step 1 - Drug Selection



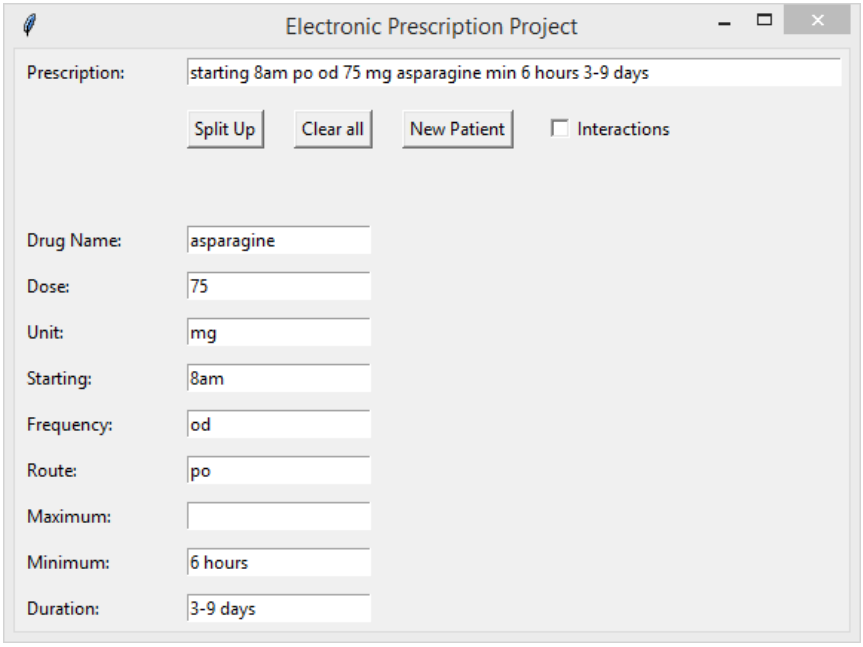
(b) Step2 - Prescribtion Details

Figure A.7: Current System

A.3 New GUI Screenshots



(a) Example 1.2



(b) Example 1.3

Figure A.8: GUI Examples from section 1.3

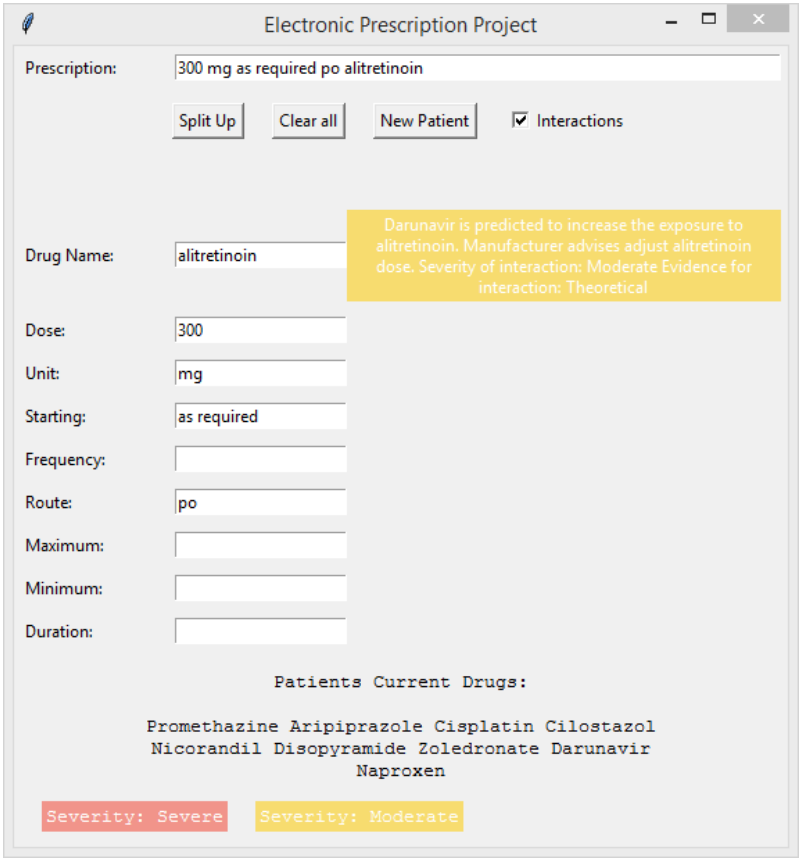


Figure A.9: GUI Interaction - Moderate Example

Appendix B

List of Abbreviations

Abbrev.	in words	Latin→ *English*
p.r.n/prn/PRN	pro re nata	as needed
o.d./OD/q.d./qd/QD	omne in die/ quaque die	once daily
b.i.d./bid/BID/b.i.s./bis/BIS	bis in die/bis die sumendum	twice daily
t.i.d./tid/TID/t.d.s./tds/TDS	ter in die/ter die sumendum	3 times a day
q.i.d/qid/QID/q.d.s./qds/QDS	quater in die/quater die sumendum	4 times a day
q.4.h./q4h	quaque quarta hora	every 4 hours
a.c./ac/AC	ante cibum	before meals
p.c./pc/PC	post cibum	after food
IV	intravenous	
PO	per os	by mouth
inh	inhalation	
IM	intramuscular	
SC	subcutaneous	
IN	intranasal	
PR	per rectum	

Table B.1: Medical Abrreviations

Appendix C

Python Code

All code for this project can be found in the following online repository:

https://github.com/olliwisly/thesis

Python code was developed using Anaconda Jupyter Notebook with Python 3.7.2. The anaconda environment file is available in the online repository.

Bibliography

[1] Samadbeik M, et al. A Copmarative Review of Electronic Prescription Systems: Lessons Learned from Developed Countries. Journal of Research in Pharmacy Practice. 2016;6.

[2] Koppel R, et al. Role of computerized physician order entry systems in facilitating medication errors. JAMA. 2005 3;.

[3] Ekedahl A. Problem prescriptions in Sweden necessitating contact with the prescriber before dispensing. 2010 9;p. 174–184.

[4] National Information Board: Paperless 2020;. [Accessed: 2019-12-12]. https://digital.nhs.uk/news-and-events/news-archive/2016-news-archive/national-information-board-paperless-2020.

[5] NHS Long Term Plan;. [Accessed: 2019-12-12]. https://www.longtermplan.nhs.uk/.

[6] Joint Formulary Committee. British National Formulary (online) London: BMJ Group and Pharmaceutical Press;. [Accessed: 2019-11-20]. http://www.medicinescomplete.com.

[7] Samadbeik M, et al. A Theoretical Approach to Electronic Prescription System: Lesson Learned from Literature Review. 2015 11;.

[8] Olufunmilola A, Chui M. E-prescribing: Characterization of Patient Safety Hazards in Community Pharmacies Using a Sociotechnical Systems Approach. BMJ quality safety. 2013 05;22.

[9] Mohsin-Shaikh S, Furniss D, Blandford A, Mcleod M, Ma T, Beykloo M, et al. The impact of electronic prescribing systems on healthcare professionals' working practices in the hospital setting: a systematic review and narrative synthesis. BMC Health Services Research. 2019 10;19:742.

[10] Gagnon MP, Nsangou ER, Payne-Gagnon J, Grenier S, Sicotte C. Barriers and facilitators to implementing electronic prescription: A systematic review of user groups' perceptions. Journal of the American Medical Informatics Association : JAMIA. 2013 10;21.

[11] Doan S, Bastarache L, Klimkowski S, Denny J, Xu H. Integrating existing natural language processing tools for medication extraction from discharge summaries. Journal of the American Medical Informatics Association : JAMIA. 2010 09;17:528–31.

[12] Wong A, Plasek J, Montecalvo S, Zhou L. Natural Language Processing and Its Implications for the Future of Medication Safety: A Narrative Review of Recent Advances and Challenges. Pharmacotherapy: The Journal of Human Pharmacology and Drug Therapy. 2018 06;38.

[13] Despotou G, Korkontzelos I, Matragkas N, Bilici E, Arvanitis T. Structuring Clinical Decision Support Rules for Drug Safety Using Natural Language Processing. vol. 251; 2018. .

[14] David S. Babtista, Hidden Markov Model and Naive Bayes relationship;. [Accessed: 2019-11-27]. http://www.davidsbatista.net/blog/2017/11/11/HHM_and_Naive_Bayes/.

[15] Jurafsky D, Martin JH. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR; 2000.

[16] Huang X, Acero A, Hon HW. Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR; 2001.

[17] Bird S, Klein E, Loper E. Natural Language Processing with Python. 1st ed. O'Reilly Media, Inc.; 2009.

[18] Matthew Honnibal, A Good Part-of-Speech Tagger in about 200 Lines of Python;. [Accessed: 2019-11-29]. https://explosion.ai/blog/part-of-speech-pos-tagger-in-python.

[19] Collins M. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002). Association for Computational Linguistics; 2002. p. 1–8. Available from: https://www.aclweb.org/anthology/W02-1001.

[20] Spoustová Dj, Hajič J, Raab J, Spousta M. Semi-Supervised Training for the Averaged Perceptron POS Tagger. In: Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009). Athens, Greece: Association for Computational Linguistics; 2009. p. 763–771. Available from: https://www.aclweb.org/anthology/E09-1087.

[21] Refaeilzadeh P, Tang L, Liu H. Cross-Validation. Encyclopedia of Database Systems. 2009 01;532–538:532–538.

[22] Niu M, Li Y, Wang C, Han K. RFAmyloid: A Web Server for Predicting Amyloid Proteins. International journal of molecular sciences. 2018 07;19.

[23] Koço S, Capponi C. On multi-class learning through the minimization of the confusion matrix norm. CoRR. 2013;abs/1303.4015. Available from: http://arxiv.org/abs/1303.4015.

[24] Labatut V, Cherifi H. Accuracy Measures for the Comparison of Classifiers. CoRR. 2012;abs/1207.3790. Available from: http://arxiv.org/abs/1207.3790.

[25] Sokolova M, Lapalme G. A Systematic Analysis of Performance Measures for Classification Tasks. Inf Process Manage. 2009 Jul;45(4):427–437. Available from: http://dx.doi.org/10.1016/j.ipm.2009.03.002.

[26] Burmater DE, Murray DM. A Trivariate Distribution for the Heigh, Weight, and Fat of Adult Men. Risk Analysis. 1998;18/4.

[27] Baba Y, Suzuki H. How Are Spelling Errors Generated and Corrected? A Study of Corrected and Uncorrected Spelling Errors Using Keystroke Logs. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Jeju Island, Korea: Association for Computational Linguistics; 2012. p. 373–377. Available from: https://www.aclweb.org/anthology/P12-2073.

Electronic Prescription: Using Natural Language Processing to improve Drug Prescription Processing

supervised by Anthony Hunter (UCL), Jonathon Wong(UCLH)

MSc in Data Science and Machine Learning, University College London

September, 2019

Current electronic prescribing systems are difficult to learn and require intensive training for every doctor, nurse and pharmacist. Even for the most basic prescriptions the process of writing and dispensing is very time consuming. The difficulty and complexity in using the software results in decreased efficiency while the need for new equipment increases costs. In Great Britain current software requires a prescriber to navigate multiple screens, fields and drop-down menus in order to complete a single prescription, and this difficulty of use is a major disincentive for end users (doctors, nurses and pharmacists).

In the thesis I discuss how machine learning can support electronic prescription processing to make prescribing faster and user friendly. In addition my goal is to improve the currently used prescription process by implementing a tagger function, a basic recommender system and a warning function highlighting severe drug interactions in the UI.

First, to prevent confidentiality issues with patient specific data I synthesized a data set of reasonable prescriptions. To synthesize mock free text prescriptions, I web scraped the British National Formulary (BNF) website, which is UKs pharmaceutical reference book/website. To generate this wide-ranging, comprehensive data set I used regular expressions to identify words from nine different classes that together form a complete prescription. These classes included the drug name, drug dose, unit, frequency, route, starting time, duration, maximum and minimum. To form a prescription the classes have been randomly ordered and 4 types of noise were added to simulate different user habits and errors.

Secondly, I tried to find the most accurate prescription classification model out of three tagging models including the first-order HMM Tagger, an Averaged Perceptron Tagger and an Bigram-Tagger. The best way to evaluate language model performance is called extrinsic evaluation, which embeds it in an application and measures its improvement. This was done by using 10-fold cross validation and performance measurements including the confusion matrix, OSR, F-statistics and many others. These measured showed that the averaged perceptron model is slightly better than the bigram model with a near eye confusion matrix and the highest OSR of 0.958.

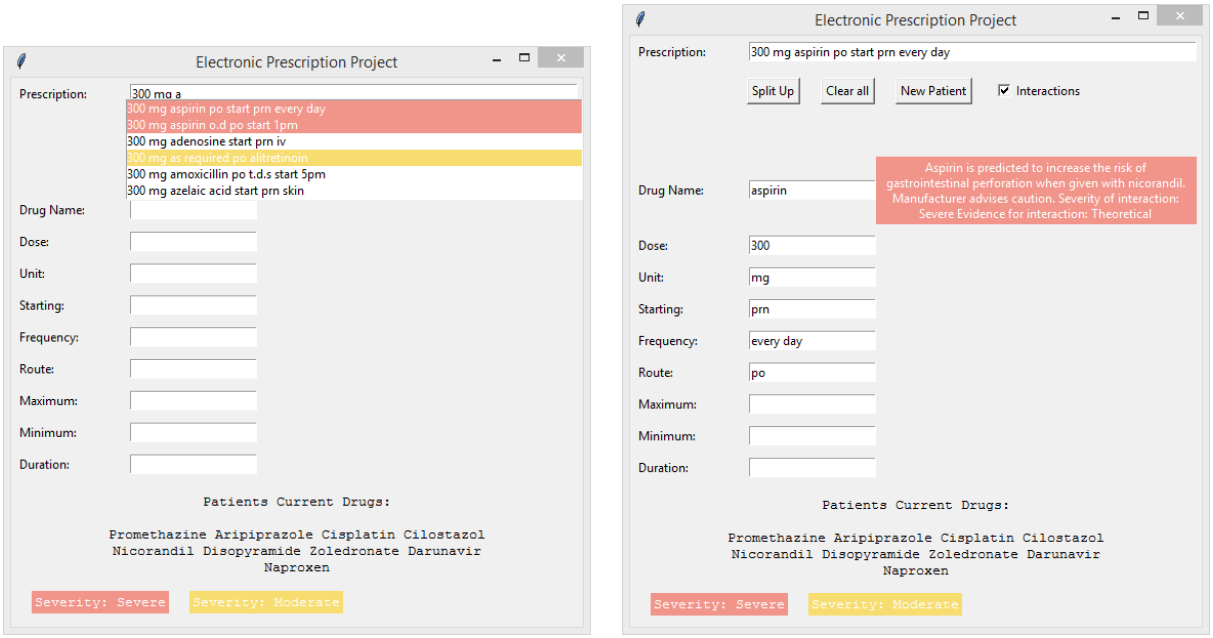
The system should not only be able to classify the final prescription correctly, but also autocomplete and recommend prescriptions while the user creates a new input to the system. The suggestion list will be built up by checking the input for possible spelling mistakes and suggesting its correction, autocompletion of the last input word in combination with its class prediction and finally forming a list of maximum 6 prescription suggestions.

How a prescription will be processed in a UI is one of the most important components in

electronic prescriptions, and was the initial motivation to write the thesis. To demonstrate the new prescription process and the power of natural language processing behind it, I developed a GUI as can be seen in example figures 1.
In a later step of the project I used the general assumption of humans being creatures of habit and rarely change their routines, hence users stick to a specific class order. Resulting in better performance for the Averaged Perceptron model from OSR 0.958 to 0.979, this increase is proportionally similar to the F-Score of these two models.

The python code of this project can be found in the following online repository:

https://github.com/olliwisly/thesis



(a) suggestion list

(b) final split and interaction details

Fig. 1: Example with severe interactions

UCL

LONDON'S GLOBAL UNIVERSITY

OLIVER WESELY

having satisfactorily completed the approved course of study and the prescribed assessment has this day been awarded the degree of

Master of Science

in

Data Science and Machine Learning

with

Distinction

Date of award: 1 May 2020

Professor Michael Arthur
President and Provost
University College London

000000172300-1

UNIVERSITY COLLEGE LONDON
LONDON'S GLOBAL UNIVERSITY

UCL

Academic Transcript

Personal Information

Student:	Oliver Wesely		
Date of Birth:	19th May 1993	University Reference:	18057603/1
HESA Reference:	1811490576031		

Programme Information

Teaching Institution:	University College London	Language of Instruction:	English
Programme of Study:	MSc Data Science and Machine Learning		
Qualification Sought:	Master of Science	FHEQ Level:	7
Mode of Attendance:	Full-time		

Award Information

Qualification Awarded:	Master of Science in Data Science and Machine Learning		
Classification:	Distinction		
Date of Award:	1st May 2020	Awarding Institution:	University College London

Module Information

Academic Year	Module Code	Module Title	UCL Credit Awarded	ECTS Credit Awarded	Result Mark	Result Grade	Attempts Completed
2018/19	COMP0080	Graphical Models	15.00	7.50	67.80	P	1
	COMP0081	Applied Machine Learning	15.00	7.50	60.25	P	1
	COMP0082	Bioinformatics	15.00	7.50	58.95	P	1
	COMP0088	Introduction to Machine Learning	15.00	7.50	76.04	P	1
	COMP0089	Advanced Deep Learning and Reinforcement Learning	15.00	7.50	77.19	P	1
	COMP0091	Project	60.00	30.00	70.00	P	1
	COMP0118	Computational Modelling for Biomedical Imaging	15.00	7.50	80.40	P	1
	COMP0137	Machine Vision	15.00	7.50	81.62	P	1
	STAT0032	Introduction to Statistical Data Science	15.00	7.50	64.92	P	1
Total credits gained:			**180**	**90**			

University Reference: 18057603/1

END OF TRANSCRIPT

Wendy Appleby

Wendy Appleby – Registrar

RAUM FÜR AMTLICHE VERMERKE DER BEHÖRDE
RESERVED FOR THE AUTHORITY

Typ/Type
P

Kode/Code
AUT

Pass-Nr./Passport No.
U 4331564

1. Name/Surname
WESELY

2. Vorname/Given name
OLIVER

3. Geburtsdatum/Date of birth
19.05.1993

4. Staatsangehörigkeit/Nationality
ÖSTERREICH

5. Ausstellungsdatum/Date of issue
13.08.2019

6. Geburtsort/Place of birth
LINZ

7. Gültig bis/Date of expiry
12.08.2029

8. Geschlecht/Sex
M

9. Größe/Height
187 cm

10. Behörde/Authority
MAGISTRAT WIEN
MBA 6/7

11. Unterschrift/Signature

P<AUTWESELY<<OLIVER<<<<<<<<<<<<<<<<<<<<<<<<

U4331564<8AUT9305199M2908122<<<<<<<<<<<<<<8