Recognising and Classifying Unique Identifiers in Unstructured Text as Named Entities Using NLP: Integrating LSTM and CNN Layers for Entities with Limited Structure and Context

BSc. Graduation Thesis

Krijn van der Burg

15th June, 2020

Submitted in partial fulfillment of the requirements
for the degree of BSc. Data Science.

at

Fontys University of Applied Sciences.

COLOPHON

This thesis was typeset using LaTeX and the `memoir` documentclass. It is based on Aaron Turon's thesis *Understanding and expressing scalable concurrency*[1], itself a mixture of `classicthesis`[2] by André Miede and `tufte-latex`[3], based on Edward Tufte's *Beautiful Evidence*.

The bibliography was processed by Biblatex. The majority of the graphics and plots were made with PGF/TikZ.

[1] https://people.mpi-sws.org/~turon/turon-thesis.pdf

[2] https://bitbucket.org/amiede/classicthesis/

[3] https://github.com/Tufte-LaTeX/tufte-latex

Afstudeerstage voor Fontys hogeschool ICT

GEGEVENS STUDENT

Naam en voorletters student: Krijn R. van der Burg

Studentnummer: 3076458

Afstudeerrichting: Applied data science

Afstudeerperiode datum van: 10-02-20 t/m 30-06-20 (90)

GEGEVENS BEDRIJF

Naam bedrijf/instelling: Ministerie van Binnenlandse Zaken

Afdeling: IT ontwikkeling & beheer

Plaats: Den Haag

Naam en voorletters bedrijfsbegeleider: F. VELDHUIJZEN

Functie bedrijfsbegeleider: afdelingshoofd

GEGEVENS DOCENTBEGELEIDER

Naam en voorletters docentbegeleider: Simona M. Orzan

GEGEVENS AFSTUDEERPROJECT

Titel scriptie: Recognising and classifying unique identifiers

Datum uitgifte: 15 Juni 2020

Getekend voor gezien door bedrijfsbegeleider:

15-6-2020

Datum

[illegible]

De bedrijfsbegeleider

If you torture the data long enough, it will confess.

—Ronald Coase

Abstract

Information can be exchanged in many different ways, the standard format for textual information is a document, which comes in an impossible number of different structures. The Ministry of Internal Affairs and Kingdom Relations concerns an overwhelming number of documents on a daily basis and uses a rule-based approach to find and capture the phone numbers, e-mail addresses, and other contact details in a given document. This brings structure to the text and as a result, quickly find the necessary information, especially because The Ministry receives an expansive number of different documents. The named entities in these documents, however, are recorded in many different formats, which made them increasingly difficult to capture as exceptions arose. I. e. the format of a named entity that does not conform, or deviate too much, from its internationally established format can not be found using traditional rule-based methods without requiring too much manual work. The hypothesis was that a machine learning approach could automatically learn and capture new formats, specifically due to its ability to adapt and generalise on new data. The first challenge to overcome was the required training data for the Machine Learning (ML) model, no internal data was made available for privacy reasons. Using scraping, textual data related to the individual named entities were collected and diversified using the data augmentation techniques: *synonym replacement*, *random insertion*, *random swap*, *random deletion*. The named entities were annotated using look-up tables and regular expressions as absolute verifiable truths. This guaranteed that the vast majority of named entities in the dataset were annotated. To recognise and classify the named entities in texts, both contextual and syntactical features were considered. The concerned named entities have a strict pattern that can mostly be identified by syntactical features. If a named entity does not conform to its respective pattern then contextual features will help in recognising, and primarily classifying, the named entity by understanding its context. A bi-LSTM-CNN-CRF model was developed as a solution. The bi-LSTM core architecture considers the context from both directions, as a result, it understands whether the context relates to a named entity. The convolution layer takes a matrix representation of individual tokens and applied several filters and a maxpool, resulting in different feature maps with syntactically captures the patterns, or the format, of a potential named entity. The CRF layer adds extra self-learned constraints to better classify the structure of a separated multi-spanning token named entity. The model achieved an average F_1 score of 0.95 on the testing subset, whereas the rule-based approach achieved an average 0.88 F_1 score. The model is better in recognising and classifying the edge-cases, as was the primary goal, but not for the base cases. Therefore, the newly developed solution should be combined with the rule-based approach rather than replace it. To create a uniform encompassing model, the emphasis should be on considering the syntactical features to a higher degree and expanding the number of syntactical features.

Samenvatting

Informatie kan in veel vormen worden uitgewisseld, voor geschreven tekstuele informatie is dit een document, wat in een onmogelijk groot aantal verschillende structuren voorkomt. Het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties ontvangt en behandelt dagelijks een overweldigend aantal documenten en gebruikt een regels gebaseerde aanpak om de telefoonnummers, en andere contactgegevens in de documenten te vinden. Dit geeft structuur aan de tekst, waardoor de benodigde informatie snel te vinden is, vooral omdat het Ministerie een groot aantal verschillende documenten ontvangt. Het probleem is dat de desbetreffende entiteiten in veel verschillende formaten worden gevonden, waardoor ze steeds moeilijker te vinden waren via de gedefinieerde regels. Dat wil zeggen dat het formaat van de desbetreffende entiteiten teveel afwijkt van het internationaal afgesproken formaat, waardoor de gedefinieerde regels handmatig constant moeten worden aangepast. De hypothese was dat een machine learning aanpak de afwijkende entiteiten automatisch kan leren en herkennen, met name omdat een machine learning aanpak zich kan aanpassen en generaliseren op nieuwe data. Er kon geen interne data beschikbaar worden gesteld, om een machine learning model te trainen, vanwege privacy overwegingen. Het gevolg was dat er een dataset gemaakt moest worden. Door middel van scraping werden teksten, gerelateerd aan de individuele betreffende entiteiten, verzameld en gediversifieerd met behulp van de technieken: synoniemvervanging, willekeurig woord invoeging, willekeurig woord vervanging, en willekeurig woord verwijdering. De betreffende entiteiten werden geannoteerd door look-up tabellen en regular expressions te gebruiken als onbetwistbare waarheid. Dit garandeert dat een meerderheid van de betreffende entiteiten in de dataset kon worden geannoteerd. De betreffende entiteiten hebben een strikt patroon waardoor de entiteiten grotendeels herkend kunnen worden via syntactische kenmerken. Als een entiteit niet in zijn geheel overeenkomt met het respectievelijke patroon, dat kan het gebruik van contextuele kenmerken helpen bij het herkennen, en voornamelijk het classificeren, van de desbetreffende entiteit. Om zowel contextuele en syntactische kenmerken te overwegen was een bi-LSTM-CNN-CRF model ontwikkeld. De bi-LSTM kernarchitectuur kijkt naar een gegeven context van beide richtingen, het begrijpt of een context betrekking heeft op een betreffend entiteit. De convolution laag

neemt een matrixrepresentatie van individuele tokens, en past een aantal filter en een maxpool toe. De uitkomst is een aantal feature maps die syntactisch alle patronen en het formaat van een potentieel entiteit beschrijft. De CRF laag voegt extra voorwaarden toe om betreffende entiteiten, die opgesplitst waren in meerdere tokens, in volledigheid te herkennen. Het model behaalde een F_1 score van 0.95 op de testsubset, terwijl de regel aanpak een 0.88 F_1 score behaalde. Het nieuwe model is beter in het herkennen en classificeren van edge-cases, maar niet in base-case. Het advies is daarom om de twee verschillende methodes te combineren. Om een uniform omvattend model te maken, wat zowel base-cases als edge-cases kan herkennen, zullen er meer syntactische kenmerken worden gemaakt en ook sterker mee worden gewogen.

Acknowledgement

The Ministry of Internal Affairs and Kingdom Relations was a great and educative environment to conduct my thesis research, consistently offering me structure and guidance, which directly improved my work and also myself. Even during a pandemic my colleagues and mentor reserved my needs and requirements on an equal footing as that of anybody within The Ministry. They changed my view on working for the government and opened a new world from where a lot of good can be done without corporate profit being the motivator. I wish them all good fortune, just as they provided to me.

Contents

List of Algorithms

List of Figures

List of Tables

Acronyms

ML	Machine Learning	vii
DL	Deep Learning	4
NER	Named Entity Recognition	xviii
NLP	Natural Language Processing	3
LSTM	Long Shot Term Memory	xviii
KNN	k-nearest-neighbour	54
TFIDF	Term frequency - inverse document frequency	34
POS	Part-of-speech	xviii
NN	Neural Network	23
CNN	Convolutional Neural Networks	xviii
RNN	Recurrent Neural Network	xviii
FNN	Feed-forward Neural Network	7
CRF	Conditional Random Field	xx
CFG	Context-free grammar	45
POC	proof of concept	

Part I

PROLOGUE

1 Introduction

The importance of automatically extracting import information from texts is rapidly increasing with the rise of social media, digital news, and blogging. Information extraction creates structure in unstructured data, allowing the data to be analysed and processed. It is a submodule of Natural Language Processing (NLP). and serves multiple purposes: categorizing articles in defined hierarchies, enabling content discovery by tagging key details, , quickly summarizing documents when time is of the essence, and revealing important information such as persons, organisations, and locations.

The idea of automatic information extraction from texts comes from the time when the first documents were digitized. But, processing texts at the time was too computationally expensive and the results were not promising[GDIV09; GE08]. However, in 2019 the *ImageNet moment for NLP* arrived with the publication of language models ELMo [Pet+18], ULMFiT [HR18], BERT [Den+10], and . The models made quick headlines by demonstrating that pre-trained language models can produce state-of-the-art results on a wide range of NLP tasks. Considering the large number of newly released NLP articles, aforementioned NER machine learning models, one can notice a trend in the exploration of new NLP techniques.

According to a study by the International Data Corporation, the amount of data grew by a factor of 44 between 2009 and 2020[GR10]. Automatic information extract can help manage this growth.

[GDIV09] Goyal et al., "Streaming for large scale NLP: Language Modeling"

[GE08] Goldberg and Elhadad, "splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications"

ImageNet [Den+10] is an image dataset organized according to the WordNet hirarchy.

[Pet+18] Peters et al., "Deep contextualized word representations"

[HR18] Howard and Ruder, "Universal Language model Fine-tuning for Text Classification"

[Den+10] Deng et al., "ImageNet: Constructing a large-scale image database"

[GR10] Gants and Reinsel, *The digital Universe Decade, Are you Ready?*

▸ Automatic information extraction can be divided into two categories: handcrafted rules and learning based methods. One approach is not necessarily better than the other, solving different problems and serving different use cases.

Handcrafted rules are generally either regular expressions, used to capture character sequence patterns; or inference code to detect syntactic patterns. This approach is generally used to capture specific patterns which can be predefined and require less generalisation. This, however, means that a single change in the respective pattern requires the system to be updated by a developer.

Learning based methods take a generic approach to capturing the patterns which makes it more flexible. ML is a field that falls under learning based methods. It is focused on creating algorithms that make predictions and classifications based on the given data. Specifically, ML aims to learn the function that maps the input data onto the desired output data. The most essential part to a machine learning solution is the training data, the samples it learns from. The patterns the model needs to learn must be annotated in the text, which can be very costly. Either in the number of man-hours it costs to annotate the data, or buying an already annotated dataset. Semi supervised approaches have been suggested to avoid part of the annotation effort[LW09][Not+13].

[LW09] Lin and Wu, "Phrase clustering for discriminative learning"

[Not+13] Nothman et al., "Learning multilingual named entity recognition from Wikipedia"

Machine Learning solutions are increasing in popularity because the model requires fewer maintenance. The generic approach to capturing patterns can cover edge cases. And more annotated datasets are becoming open-source which reduces the cost.

▸ Named Entity Recognition (NER) is the process of identifying specific elements and groups of elements which share common semantic characteristics. *Named entities* can be defined as one or many character sequences which stand consistently for some referent[1]. Previously, NER relied on a large knowledge base, called an ontology. The result of the NER model can be very generic or domain specific, depending on the ontology. E. g. an encyclopedia of all branches would need a high-level ontology to structure and understand all the data, in contrast, a specific domain journal needs a detailed ontology due to the complexity and number of similar terms[2] [SCAG13]. The downside of ontology based NER is that the model requires constant updates to keep up with the ever-growing available knowledge which creates new terms on its own.

[1] The term *named entity* is loosened for various types of entities such as temporal expressions and numerical expressions, e. g. the year *2020* points to the *2020th year of the calender* while *February* refers to a month of an undefined year.

[2] An NER model in the medicine and healthcare domain would require a high-level ontology to understand the differences between all complex medical terms.

[SCAG13] Sanchez-Cisneros and Aparicio Gali

Machine Learning solves this problem and is more precise due to its ability to cluster words. However, ML requires the text to be processed into features for it to understand text. And still it would have a difficult time understanding the semantic and syntactic relations between words. A subfield of ML, called Deep Learning (DL), solves this problem using a technique called word embedding, changing words into vectors where related words take similar vector values, occupying the same embedding space. DL also offers many different techniques and algorithms to best tackle the NER problem. The architecture of a DL model can be continuously improved and build upon with new and different DL layers. For example, an LSTM algorithm will only be able to use past information of a sequence in the text to make a prediction, while a bi-directional LSTM can understand the past and future label of a sequence as it processes the text from left-to-right and right-to-left. Additionally a CNN layer can convolve a sentence into different feature maps which might capture top-level patterns. A benefit of DL is that feature engineering is not required as much. Only labelled, or annotated, data is required, in combination with an embedding layer.

1.1 THE PROBLEM

Unique identifiers, such as phones and e-mail addresses, are generally extracted via handcrafted rules using regular expressions, since these entities have an (internationally) established standard format. However, these entities are not always documented in this established format[3]. Most of these cases can still be captured properly by rule based methods. Unfortunately, as the amount of data grows, the number of edge cases also grow[4]. These edge cases are increasingly difficult to capture. Regular expressions are rule based and inherently very strict on the format, the expression can be made more generic which results in more false positives. Or a regular expression could be made for each edge case, resulting in an increasing amount of maintenance. Everything can be captured by a regular expression, the problem is creating and maintaining said expressions on an increasing number of edge cases.

[3]Phone numbers have an international standard (e. g. a + to indicate an international number, followed by the country code, followed by the local number), however, number sequences are often written with spaces, hyphens, or slashes in-between numbers to make it easier to read (e. g. $+00 - 0000 - 0000$, $+00/0000/0000$).

[4]The following number sequence 06.062 34 256 is not directly recognizable. A person, however, can extrapolate from context that the number sequence concerns a phone number.

A learning based approach, such as a NER Machine Learning model, could solve this problem. However, current state-of-the-art models are trained on an expansive range of domains with a broad vocabulary, only capable of recognising entities registered in MUC-7[CR98]. Unique identifiers, are not recognized by these models. The new entities could be taught to the ML model via transfer learning. However, unique identifiers are not annotated in any publicly available corpora. Annotating such dataset would be too costly. Furthermore, unique identifiers have no linguistic properties. Making them increasingly reliant on context to be recognized. Many unique identifiers are also solely numerical and belong to the *information, communication, and technology* domain. Sharing very similar contexts which makes it increasingly difficult for NER algorithms to relate a distinct context to a specific unique identifier.

[CR98] Chinchor and Robinson, "Appendix E: MUC-7 Named Entity Task Definition (version 3.5)"

1.2 MY THESIS

The thesis opens by deepening our understanding of Named Entity Recognition through literature study and exploring related works to support the thesis. The goal of this thesis was to improve the state of affairs, as described in chapter §1.1, along three axes. First, developing new ways to automatically collect quality textual data into a dataset and annotate the containing unique identifiers. Second, designing and developing a new ML model that can distinguish unique identifiers in documents by both considering the context and the format of the named entity. To that end, this thesis demonstrates two claims, elaborated upon in turn:

A. Named entities in a large unlabelled corpus can automatically be annotated into a distilled subset using regular expressions, lookup tables, and gazetteers.

B. Unique identifiers can be recognised and classified in a given documents while deviating from its established format using the generalisation and adaptability factors of Machine Learning.

1.2.1 *Automatically collect and annotate a corpus*

A substantial number of annotated corpora can be found publicly available, under a free license. Often, however, the annotated named entities are exclusively MUC-7 registered [CR98]. This research concerns itself with recognising and classifying unique identifier entities such as phone numbers, e-mail addresses, and bank account numbers. Unfortunately, said entities are not annotated in any publicly available corpus, or only to an unusable amount. Therefore, a new dataset had to be created, and due to time and budget limitations this had to be done automatically.

[CR98] Chinchor and Robinson, “Appendix E: MUC-7 Named Entity Task Definition (version 3.5)”

This thesis claims that an extraordinary large corpus, such as the *enwiki dump* or *google news* collections, can be extracted or distilled using gazetteers, lookup tables, or regular expressions, into a subset, where each of the concerned named entity is automatically annotated. A restrictive algorithm[5] can select all the named entities without introducing false positives or true negatives while guaranteeing the accuracy of true positives. Reducing the number of true negatives in an NER dataset does not affect the quality of the data, as the number of true negatives is more than tenfold the number of true positives. False positives can be prevented by checking each entity to an absolute truth[6]. False negatives are more difficult to prevent, but only arise in more exceptional cases. A selection algorithm could include a false negative recognised named

[5] E. g. regular expressions on a specific format or inference code that requires any recognised person name to be in a lookup table or otherwise its discarded.

[6] E. g. a phone number that is confirmed by both a regular expression and look-up table is confirmed as a phone number named entity.

entity as the entity resides close to another true positive named entity[7]. The accuracy loss due to true negatives would be insignificant, marginal at best.

[7]E. g. a context containing two named entities, one true positive and one true negative, would be captured in its entirety, while the true negative entity remains unlabelled.

A corpus labelled by computers is called the *silver standard*, as the common census is that a computer misses a number of named entities which a person would have caught. However, this thesis claims that a computer can create and annotate a corpus with a precision and recall score similar to a human performance using regular expressions, look-up tables, gazetteers, or a combination.

1.2.2 *Recognise and classify unique identifier named entities*

Regular expressions often work great to capture unique identifiers from text. This is because unique identifiers are only unique in its value, not its pattern. Unique identifiers such as phone numbers and e-mail addresses have an internationally established format it should conform to. As stated in chapter §1.1, this thesis focuses on capturing unique identifiers which were not documented conform its format. A simple method would be to loosen the regular expressions, however, this would introduce an unacceptable amount of false positives[8].

[8]E. g. a large number that is not divided by punctuation marks could be recognised as both a cardinal and phone number

Another strategy is to capture the named entities by its context. If a certain combination of words refer to a phone number, and the words point to a sequence of numbers divided by punctuation marks, then that must be a phone number. However, unique identifiers are often found in short texts with similar contexts[9].

[9]E. g. the words used to reference a phone number and email can be the same, such as in "*my contact details are..*" and "*reach me on..*"

This thesis presents three different ML models to capture unique identifiers in a given document, considering both the format and the context to maximise its effectiveness.

The first model uses a bi-directional LSTM to consider the context surrounding the current token and a CNN to create feature maps that capture the format of said token by convolving the characters. Additionally the casings of the token is supplied as an encoded vector .

[9]E. g. whether the word is capitalised, lowercase, uppercase, or contains a specific punctuation mark such as an at sign.

The second model uses a two-stage Feed-forward Neural Network (FNN) architecture. The first FNN model serves to recognise the named entities, it outputs a list of token subsequences, which it scores as likely named entities, called candidate entities. Allowing the second model to only focus on separating the named entities in a feature space while not having any other type of entity into account. The second FNN model classifies the tokens from the first stage FNN model. The features are a set of tokens preceding and proceeding the candidate entity, a set of binary casings, and a bag-of-words of the candidate entity separated by a space.

THE THIRD MODEL is a SpaCy NER model trained on custom data.

1.2.3 *Research questions*

Through laddering and triangulation, the following sub-research questions were formulated to establish a clear documented roadmap of this research. Approaching the project from different angles and perspective, to make sure the data and results are cross-checked via different methods.

► AUTOMATICALLY COLLECT AND ANNOTATE NER DATA

- How can data be automatically gathered to gradually or continuously add new data samples?
- How can data be automatically annotated without introducing false positives or false negatives.
- What techniques can be applied to diversify and expand existing data samples?

► RECOGNISE AND CLASSIFY UNIQUE IDENTIFIER NAMED ENTITIES

- Which annotation scheme best capture an named entity in a document?
- Do token casing features improve NER accuracy?
- Which form of embedding best capture the named entities and its context?
- Which machine learning algorithm and architecture best recognise and classify annotated entities.
- How to compare a production solution and the new developed machine learning model?
- How to automatically deploy a trained model to a production environment?
- Can a ML model size be reduced without significantly compromising its accuracy?

1.3 RESEARCH ENVIRONMENT

This research was conducted at The Ministry of Internal Affairs and Kingdom Relations of The Netherlands. Established in 1798 under the name of Department of interior police and supervision on the state, dikes, roads and waters of the Batavian Republic. The main objectives The Ministry concerns itself with are:

- Safeguarding the democratic constitutional state
- The care for a well functioning public administration
- Ensuring the quality of personnel and management in the civil service
- Guarding the constitution
- The successful participation and integration of new residents
- Urban policy and population decline
- Housing market (Purchase and rental sector, including housing corporations)

Due to the nature of the organisation and its objectives, The Ministry concerns an overwhelming number of documents. The team I am part of focuses on developing applications which automatically bring structure to these documents and process them accordingly. Reducing the required amount of manual work and improving the efficiency of The Ministry.

1.3.1 *Methodology*

This thesis was divided into five distinct overarching phases. First, all the necessary information was gathered and meetings with the stakeholders and users were held (initialise). Second, the problem was defined, main research questions are formed, and a plan to develop a solution was proposed (define). Third, the available data sources were analysed, feasibility of the project was tested, product requirements and non-functional requirements were documented, sub-questions were formed that further detail or assure the quality of the main question, and as last the plan was accepted (exploration). Fourth, the sub questions were researched and together answered the main question, and the product was developed in accordance with the established requirements and non-functional requirements (realisation). Fifth, the product documentation was finished, the product was handed over to the stakeholders, and the results were presented to the users (transfer).

The project used an agile approach, dividing the project into periods of two weeks unless a phase ended sooner. Tasks were discussed and assigned at the start of each period, at the end of each sprint there was a demonstration of the product and its results. After that, the progress was documented and a plan was made for the next sprint.

1.4 OUTLINING

This thesis is divided into three largely independent parts. The first part, *understanding named entity recognition*, summarises related works and findings and provides the theoretical background and technical perspective that informs the rest of the thesis. The second part, *automatically collect and annotate data*, explains how the data was collected, annotated, and feature engineered to train ML algorithms. The third part, *recognising and classifying unique identifiers*, details the ML models experiments and results, representing my contribution to the task of NER in unique identifier named entities.

► UNDERSTANDING NAMED ENTITY RECOGNITION

Chapter 2 describes the measures taken to assure the quality of this research and prevent common biases. Triangulation was performed as the DOT framework instructs, approaching the research from different angles and perspective, to make sure that the data and results are cross-checked via different methods.

Chapter 3 explains the dependencies and technologies used in this thesis. First, explaining the concept of NER (§3.1) and the unique identifiers (§3.2) it should capture. Second, detailing what a dataset and corpus (§3.3) is and how ML (§3.4) can learn patterns from data. Third, listing and detailing the various DL layers (§3.5 through §3.8) used throughout this research and how to speed-up training time by using transfer learning (§3.9). Fourth, explaining what regular expressions (§3.10) are and its purpose.

Chapter 4 analyses related works in NER and using ML to capture unique identifiers. First, listing all the different forms of NER (§4.1). Second, analysing recent advancements in incorporating convolutional DL layers in NLP tasks (§4.3). Third, analysing the results of a two-stage quantized FNN that recognises and classifies named entities separately (§4.4). Fourth, understanding the usage of engineering casing features in an FNN to classify number formats (§4.5).

Chapter 5 opens by explaining why text must be represented by numbers (§5.1). It is followed by detailing two different methods of representing text. First, one-hot encoding and bag-of-words as discrete representation and its limitations (§5.2). Second, using neural networks to assign a vector to embed a word to a spatial embedding space as distributed representation along with its limitations (§5.3). Last, detailing the problem with embedding unique identifier named entities (§5.5).

Chapter 6 opens by globally explaining how an NER model is evaluated and what the acceptance criteria of the final model should be (§6.1). Three different evaluation metrics are discussed, each improving on the former. First, simple accuracy score and why it is often inaccurate (§6.2). Second, mean squared error and how cross-entropy error improves on it (§6.3). Last, F_1 score as a weighted combination of the precision and recall score (§6.4).

▸ AUTOMATICALLY COLLECT AND ANNOTATE DATA

Chapter 7 lists the proof-of-concept dataset creation (§7.2) and various ML models based on related work chapters §7.3 through §7.5. The result of each model is analysed to determine if it had to be pursued further.

Chapter 8 opens by explaining the purpose of training data for NER ML algorithms, and the reasons why a custom dataset had to be created (§8.1). First, textual data was collected via scraping and the named entities were annotated (§8.2 and §8.3, respectively). Second, combining and preprocessing the data to construct positive samples (§8.4) and negative samples (§8.5) in combination with random noise (§8.6) for a balanced and wide dataset.

Chapter 9 explains the purpose of adding additional features to an NLP dataset (§9.1), detailing the features and its implementation on a per chapter basis (§9.2 trough §9.6). Last, concluding the resulting dataset and providing a list of all engineered features (§9.7).

- RECOGNISING AND CLASSIFYING UNIQUE IDENTIFIERS

Chapter 10 explains different vector reduction techniques(§10.1), while maintaining embedding accuracy. And bucketing methods (§10.2) to reduce padding lengths in training batches. Last, detailing the software implementation in chapter §10.2.

Chapter 11 describes the bi-LSTM-CNN-CRF model in detail and elaborates on the benefits of the base LSTM layer (§11.1) and features (§11.2). Each consecutive section introduces additional deep learning layers and features to the model its architecture that improve its accuracy (§11.3 through §11.5). Last, the results are concluded and analysed (§11.6).

Chapter 12 opens by explaining two-stage FNN architecture and re-iterates the related works that demonstrate and discuss this approach (§12.1). Each architectural stage is detailed on a per chapter basis (§12.2 and §12.3) and the results of bot models is aggregated, analysed, and discussed (§12.4).

Chapter 13 covers the custom training of the SpaCy NER model. First, covering the architecture as is known (§13.1), an exact re-iteration of chapter §7.5. Second, listing the results and the analysis (§13.2). Data processing steps and feature engineering steps are not covered as the model does not provide customisation options.

- EPILOGUE

Chapter 14 lists the additional steps and methods to evaluate the final bi-LSTM-CNN-CRF model besides the testing subset. First, using public datasets that contain one of the named entities and manually evaluate the results (§14.1.1). Second, comparing the former implementation and the final model by submitting the same production data to both systems and analysing the results (**??**). Last, concluding the results of both tests and discussing a combination of the results (§14.1.4).

Chapter 15 concludes the achieved results to support the thesis claims, and provides a structured roadmap to further improve the results and state of affairs (§15.1).

2

Research quality

▸ SYNOPSIS This chapter describes the measures taken to assure the quality of this research and prevent common biases. Triangulation was performed as the DOT framework instructs[1], approaching the project from different angles and perspective, to make sure that the data and results are cross-checked via different methods.

[1] `http://ictresearchmethods.nl/The_DOT_Framework`

▸ EXPERT INTERVIEW During the *defining* phase of the project, multiple interviews were held with field experts. Exploring the problem definition, as described in chapter 1.1, and discussing optimal solutions. Using the *five whys* technique, the root cause for this project was uncovered, which is about reducing the amount of manual work to maintain, create, and update the regular expressions used in the former implementation.

▸ SWOT AND RISK ANALYSIS To support the decisions made during this research, all the strengths, weaknesses, opportunities, and threats related to this project were identified. The results were combined with its occurrence chance and compiled in a matrix. Assigning a risk score to each factor along with a mitigation strategy. As a result, each encountered problem during this project was swiftly tackled and solved. Mostly it prevented time from being unnecessarily wasted. Indirectly improving the research quality, as the available time was limited.

▸ LITERATURE STUDY NLP is a relatively new field, especially its recent advancements introduced a substantial amount of new technologies, ML architectures, and techniques. The advancements are too recent, no clear and distilled explanatory or showcase material exists yet. As a result, an immense number of papers and articles from published journals had to be read and analysed. There were cases where the documented mathematics was beyond my understanding. This was distilled and explained to my by colleagues with a mathematics degree. Allowing me to understand, and thus implement, the state-of-the-art technologies and architectures. Directly improving the quality of this research and in turn the results.

- Interview Users of the former implementation were interviewed top gain a better understanding of the defined problem. No new insights were gained as the only request by the users is a more accurate system. A notable preference is that all the named entities should be found, classifying the named entity to it is properly label is a secondary concern. For the users it is easier to correct a mislabelled entity rather than finding entities in a given document themselves. This directly relates to the recall score used to determine how many named entities were recognised. However, this does not mean that the named entities were classified correctly, that is indicated by the precision score. More details on the evaluation of an NER ML model is documented in Chapter 6.

- Observation The former implementation was demonstrated to gain a better understanding on what the result should look like, and what the users expect as output. The result of the new implementation must remain the same as other systems are dependant on the output format, which present the output to an understandable format. Additionally, the performance of the former system was demonstrated and its hardware and technical details shared. To be used as minimal performance requirement of a new implementation, and as a baseline to strive for further performance improvements.

- Document analysis After receiving a demonstration of the former implementation, all the available documentation was analysed to better understand the use case. As the access to internal data was restricted, the documentation helped shape the input data and output format. Additionally helping during the *defining* phase of the project by demarcating the functional requirements and non-functional requirements. This information was predominantly processed and documented in the project plan (appendix refappendix:project plan)

- Data analytics To get a firm grasp and understanding of the data used in this research, a substantial amount of time was spend on analysing potential data sources and the data itself. Analysing the data resulted directly in additionally engineered features, which significantly improved the final ML models results. A number of visualisations were made to support the analysis and convey the information as clearly as possible, these are depicted in its respective chapters.

- (non-) functional test To determine whether the newly proposed solution, concluded from this research, surpasses the former implementation, a series of functional requirements and non-functional requirements, established in the project plan, must be met. Primarily the accuracy of the newly proposed solution compared to the former implementation, this validation data is further detailed in prologue chapter §14.1.1. The performance of the newly proposed solution is measured via various metrics, such as system load per given document related to its execution time. These metrics are directly compared to the former solution, determining whether the newly proposed solution is on par.

- Product review After validating the ML model using the training data and validation data (§14.1.1), the model is deployed parallel to the former implementation. Allowing the model to be reviewed by peers and users while not limiting or damaging the current production, in case the model is faulty. For every iteration, the given feedback is collected and processed. Incrementally improving the model during the final stage until it is accepted and fully deployed.

- Benchmark test The ML models designed and created during this research are evaluated using the F_1 score, more details can be found in Chapter 6. The testing data was separated from the training data before nay form of analysis or (pre-) processing was applied. A validation set, a collection of production documents, was used as a final acceptance test. In concordance with the defined evaluation metrics, to give a final model performance score. This validation set is also applied on the former implementation to be able to directly compare the results.

- Brainstorm Every two weeks a meeting with managing stakeholders was held. The meeting concerned research progress and general product increment. On occasion, field experts were invited to these meetings to spar and brainstorm on new innovative approaches to gain the desired result. These expert meetings directly addressed two methods which had not yet been considered. The first being sequence bucketing, to reduce training time, which was implemented along with model reduction techniques. Second was to implement Bayes theorem, to provide a traceable answer on how and why the named entity was found. However, due to time limitations this was not implemented.

- Code review Any notable code increments were persisted locally and via a version control system. Using a ticketing and issue tracking system, improvements to existing architecture and production code is only added after peers reviewed and formally approved the code. This system was also used to ensure the code was secure and conform to the code style guidelines.

- Prototyping Before any approach or ML model architecture was researched and tested in its entirety, a proof-of-concept was developed and evaluated. As a result, a selection with of the most promising results was further pursued, allocating time more efficiently. Each proof-of-concept also helped shape the required data format for feature engineering and ML model training.

- Root cause analysis The problem this research aims to solve are unique identifier named entities that could not be found in the given documents. By manually labelling a select number of documents and apply the former implementation on said documents, a list was compiled of named entities that were not recognised. After analysing the unfound entities it became clear that all entities deviated from its expected format. This finding directly resulted in additionally engineered features, as described in chapter §9.3, which improved results significantly.

Part II

UNDERSTANDING NAMED ENTITY RECOGNITION

3

Task definition

▸ SYNOPSIS This chapter explains the dependencies and technologies used in this thesis. First, explaining the concept of NER (§3.1) and the unique identifiers (§3.2) it should capture. Second, detailing what a dataset and corpus (§3.3) is and how ML (§3.4) can learn patterns from data. Third, listing and detailing the various DL layers (§3.5 through §3.8) used throughout this research and how to speed-up training time by using transfer learning (§3.9). Fourth, explaining what regular expressions (§3.10) are and its purpose.

3.1 NAMED ENTITY RECOGNITION

NER detects and classifies textual sequences into a predefined set of named entities. An example of named entities could be *persons* (PER), *organisations* (ORG), or *locations* (LOC). These entities are registered under MUC-7[CR98], which is a list of named entities established during the MUC-7 conference. The output of na NER model is the same text it was given but with each named entity highlighted in some form. An example NER output is shown in figure 3.1. The type of named entities this thesis concern itself with are unique identifiers, which are not MUC-7 registered. Chapter §3.2 explains and defines unique identifiers in further detail.

[CR98] Chinchor and Robinson, "Appendix E: MUC-7 Named Entity Task Definition (version 3.5)"

$$\underset{PER}{\underline{\text{Peter}}}\ \text{works at}\ \underset{ORG}{\underline{\text{De Nederlandsche Bank}}}\ \text{in}\ \underset{LOC}{\underline{\text{The Hague.}}}$$

FIGURE 3.1: NER example output.

Each named entity has a span and a type. The span is the exact location of the named entity within the text. Named entities often consist of multiple words or character sequences, hence its called a span. In the NER example output of figure 3.1, the named entity POS spans two words. The type is the label of the named entity. Defining the type of a named entity is very dependant on the context. Marking *Nederlandsche* as a location, in the NER example output of figure 3.1, would be grossly incorrect.

3.2 UNIQUE IDENTIFIERS

The use of unique identifiers is everywhere and all around us everyday. From the *citizen service number* on the back of identification cards, phone numbers, email addresses to usernames on websites. The purpose of a unique identifier is that it is always *unique*. It allows for information to be associated only with a single entity. The foremost reasons to use unique identifiers is for security purposes and identifying objects.

Unique identifiers take many forms, each with a different generation strategy. *Serial numbers* are used mostly when knowledge of the unique identifier doesn't matter, typically incremental or sequential. *Random numbers* when security is a concern but duplicates and collisions can be dealt with, usually hashed or taken from a dedicated random process. And *names* to identify people or objects on a more natural level, can practically only be created by people.

An email address, for example, is just a unique name when it comes to UI's. The first part of an e-email address is picked by the user, generally the persons name, and the latter part is the company or service providing the email service. Where the @ symbol is only used to distinguish the two parts for identification and processing purposes.

Universally unique identifiers are just random numbers, often with an exact timestamp. Creating unique identifiers computationally is difficult. A computer is nothing but logic, it inherently cannot be random. There is always a state forcing the computer to do something, hence it's not random. People can interpret the result random or it can be deemed random enough, where collisions are of no concern. Typically, a UUID just picks a sequence of bits picked by a random number it generated. Followed by the exact current timestamp as an integer as an extra guarantee[LMS05].

[LMS05] Leach et al., "A Universally Unique IDentifier (UUID) URN Namespace"

ML algorithms are trained on a large collection of data samples, called a dataset or corpus. The following chapter (§3.3) defines a dataset and corpus, and details the differences between them.

3.3 DATASETS AND CORPORA

A collection of data is called a dataset. There are many different forms of datasets, the most common is a tabular form with rows and columns. Where each column represent the same type of data and the row every sample. NER algorithms, however, are trained on a large and often continuous collection of texts, called a corpus. Ruettet defines a corpus as a ”representative sample of *actual language production* within a *meaningful context* and with a *general purpose*. Whereas a dataset is a representative sample of a specific *linguistic phenomenon* in a restricted context and with annotations that relate to a *specific research question*”. The named entities this thesis aims to recognise are generally contained within short texts with little context and can be observed as a domain specific linguistic phenomenon. Hence the term dataset is more accurate.

A data sample of a corpus is called a document, because often a corpus is a collection of actual documents and articles. For the sake of simplicity this thesis refers to individual samples in corpora as *texts*. Since the data samples aren’t taken from documents either. More details on the dataset and technical implementations can be found in Chapter 8.

The named entities within these texts are annotated, teaching the model the linguistic properties and the context surrounding the entity. There are various methods of annotating data in text, called sequence labelling. The following chapter (§3.3.1) defines sequence labelling and details the various applicable methods.

3.3.1 *Sequence labelling*

Datasets contain columns with a single type of value representing a feature, often as a result from preprocessing or feature engineering. The only data considered in an NER problem is text, here everything is a feature, down to the character level. A corpus is just a large body of text and NER algorithms need to know what to look out for.

Since corpora are usually a single file, containing the entire texts. It’s unwise to label words by its word index position, when a single word is added to the file, the entire index shifts and the corpus must be completely re-indexed. Since named entities can also span multiple words, the named entity requires at least two indexes, a beginning and end span.

Sequence labelling assigns a class or label to a token inside the text. There are different methods of sequence labelling, the most common are the *IO*, *BIO*, and *BILOU* methods, each improving on the former. One of the first methods used was IO[BSW99; CS02]. Where the named entity sequences are labelled with an *I*, and the rest of the sequence with an *O*. First introduced by Ramshaw and Marcus, some validation corpora, such as the CoNLL sets[TKSDM03], use the BIO method. Improving on IO by specifying the start of the sequence with a *B*, the rest of the respective entity sequence is marked by an *I*. BILOU is the most extensive and complex form of labelling. Further improving on the former methods by additionally labelling the last sequence of the respective named entity with an *L*. And using *U* to label an named entity spanning a single token, mostly to increase performance. An example BILOU output is illustrated in figure 3.2, where *PHO* is the label for phone number.

[BSW99] Bikel et al., "An Algorithm That Learns What's in a Name"

[CS02] Collins and Singer, "Unsupervised Models for Named Entity Classification"

[TKSDM03] Tjong Kim Sang and De Meulder, "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition"

(**B**) Beginning is the first word of the named entity.
(**I**) Inside of the named entity, which can span multiple tokens.
(**L**) Last word of the named entity.
(**O**) Outside words that are not part of the named entity.
(**U**) Unit if the named entity spans a single token.

$$\underset{O}{\underline{\text{Peter's}}}\ \underset{O}{\underline{\text{phone}}}\ \underset{O}{\underline{\text{number}}}\ \underset{O}{\underline{\text{is}}}\ \underset{B-PHO}{\underline{\text{+31}}}\ \underset{I-PHO}{\underline{\text{06}}}\ \underset{L-PHO}{\underline{\text{12345678}}}.$$

Figure 3.2: Sequence labelling BILOU example output.

3.4 MACHINE LEARNING

There are two different approaches to creating an NER model. The rule based approach covered in this thesis uses regular expressions, defined and explained in chapter §3.10. For learning based approach this thesis covers Machine Learning, defined in chapter §3.4.

Learning based NER approaches often use a Machine Learning (ML) implementation. This thesis focuses on exploring various ML models and configurations as NER solution to the stated problem (§1.1). Machine Learning belongs to the fields of computer science and statistics. Performing specific tasks without being explicitly programmed, relying on patterns build from mathematical models based on sample data. An optimally trained model will correctly determine labels for new unseen samples. Called the generalisation and adaptability factor of a model. There are a vast amount of different ML algorithms, most which can be categorized as either *supervised* or *unsupervised* learning. NER algorithms primarily use supervised methods to learn the named entities from texts.

► Supervised learning is a method that learns a mathematical model or function that maps an input to an output based on the labelled sample data. The data that's being fed to the algorithm has the answers attached. Helping the model to recognise patterns that lead to the correct answers, an intuitive mapping from input to output. In NER this is achieved by providing a corpus with the named entities highlighted using sequence labelling.

The foremost problem with ML algorithms is that it can overfit or underfit a model. *Overfitting* means that the model associates specific input samples with output labels instead of learning a general mapping of inputs to outputs. Meaning the ML algorithm is less able to recognize training samples as it's changing constantly, resulting in smaller weights with a lower generalization error[GBC16]. *Underfitting* is, de facto, the exact opposite of overfitting. The algorithm wasn't supplied with enough data samples to properly learn a mapping. As a result it generalizes too much and, shortly said, the output can basically be considered as random. Chapter (crefch:dataset) details the methods used to prevent these problems in this thesis. A simple definition of a properly fitted model can be described as: supplying as much data on as many iterations as possible, before the error rate on the testing data increases. See figure 3.3 for an overfitting visualisation.

[GBC16] Goodfellow et al., *Deep Learning*

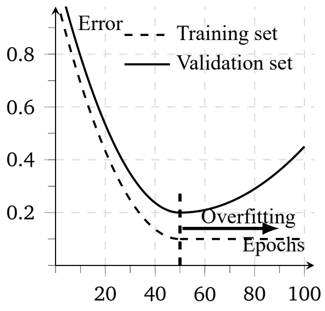


FIGURE 3.3: Machine Learning algorithm overfitting visualisation

3.5 NEURAL NETWORKS

Often compared to brains, Neural Network (NN), simulate dense and fully interconnected layers of neurons to learn how to recognise patterns. Typically, a standard neural network has a range from a few dozen to thousands of artificial neurons, also called units or neurons, arranged in a series of layers. Neurons have values ranging from 0.0 to 1.0, where 0 represents fully inactive and 1 represent fully active, with all (infinite) possible values in-between. Each neuron in each layer is connected to its neighbours layer's neurons, as depicted in figure 3.4. The lines connected from one neuron to another are called *weights*. These determine whether the value of the activated neuron should activate the connected neuron.

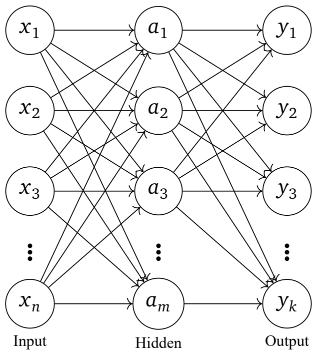


FIGURE 3.4: Neural network (multiplayer perceptron) example visualisation.

An NN consists on three types of layers. The left most layer in a NN is called the *input layer*. All the neurons in the input layer are the variables *fed* to the NN. The right most layer is called the *output layer*. The neurons in the output layer represent the classes. The amount of output values is equal to the amount of output neurons. All the layers between the input layer and output layer are called *hidden layers*, these are responsible for detecting patterns which make NN purposeful. Hidden layers detect patterns by activating neurons only if the weighted threshold of multiple other neurons from previous connected layers reached an activation threshold. Whether this threshold is reached is determined by an activation function, this is further detailed in chapter 3.7.

In a fully connected layer, a single neuron is connected to all the neurons of the previous layer. The neuron has an activation threshold, meaning that it only activates when the other connected neurons have high enough values. This process is visualised in figure 3.5. Where x_i are the neurons in the input layer, w the weights, and Σ the weighted sum.

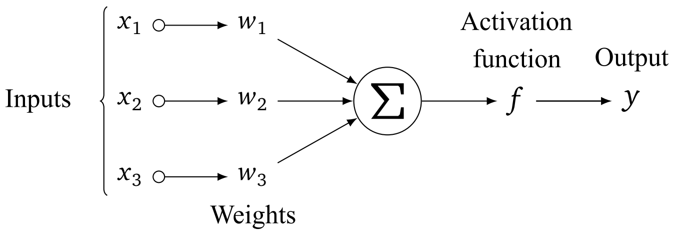


Figure 3.5: Neuron activation process visualisation.

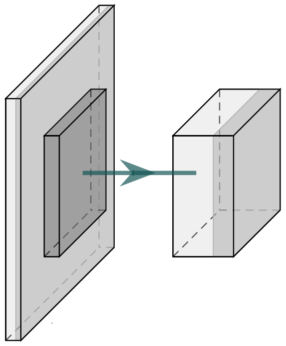


Figure 3.6: CNN input convolving to new feature output map visualisation.

3.6 CONVOLUTIONAL NEURAL NETWORKS

There is a wide variety of different NNs. Convolutional Neural Networks (CNN) differentiate themselves from *regular NNs* with special hidden layers called convolutional layers. These layers transform the data by performing cross-correlations, called convolving. A CNN convolves over the input layer, calculating the dot product between the filter and the input. The result is a new activation map output, called a *composition*. A local data map of low-level features into a high-level representation. See figure 3.6 for a visualisation. Each layer applies different filters, usually of different sizes. The combined results are new features with detected patterns based on different top level views of the data. Every neuron in the CNN is only connected to a small portion of the input map from in adjacent layers. This is called *local invariance*, the features are only considered by a respective few neurons.

Convolutional Neural Networks are often used in image recognition software. Using the convolution layers to detect shapes such as edges and circles, which together can identify an object. Recent publications also show that CNNs are also applicable on textual data.

An embedded document takes the form of a matrix. Where each row is a single embedded token and the columns the vector. If a document contains 12 tokens and is represented by a 100-dimensional embedding then the matrix shape is $12x100$. On image data a filter sliders over the enitre image, top to bottom and left to right. For NLP a filter usually slides over full rows of a matrix. Generally the window slides over 2 to 5 words at a time.

Even though words in a sentence can be semantically related, this doesn't mean that they're part of the same location invariance. Two related words can be relatively far away from each other within a sentence, separated by several words. This could indicate that CNNs aren't a good fit for any MLs related task. However, combining a CNN with an RNN makes more sense. Resembling how language is supposed to be read, sequentially from left to right. RNNs are further detailed in chapter 3.8.

3.7 ACTIVATION FUNCTIONS

Neural Network have a number of hyper-parameters which affect the algorithm it's training and testing behaviour. The activation function hyper-parameter calculate whether the weighted sum meets the neuron its threshold. If the threshold is met then the neuron is active which serves as input to the subsequent layers.

The three main activation function this thesis uses are the *Sigmoid*, *ReLu*, and *Softmax* functions. The SIGMOID function takes a value between -1 and 1, often used for binary classifications. RELU is often used in hidden layers, improving on other activation functions with a reduced likelihood of the vanishing gradient problem. During training and back-propagation the neurons receive an update to update its weights. If the update value is too small then the weights don't change, RELU minimizes this problem. SOFTMAX turns a numeric output of multinomial NN output layer into probabilities for each classification output. See table D.1 in appendix D for a list of activation functions and its value range. The other listed activation functions are variations on the discussed three activation functions. In practice only two activation functions are used per model. One for the input layer and all hidden layer and another for a classification or probability distribution over the possible classes.

3.8 RECURRENT NEURAL NETWORKS

Most ML algorithms consider one data sample at a time, which can consist of multiple features. Time-series data, however, is related to its previous data sample and next sample input. In textual data, the position of the word is the time factor, each word in a sentence is in some form related to its neighbouring words. An common NN, however, cannot relate the different time steps to one another, Recurrent Neural Network (RNN) solves this problem. An RNN is an NN with feedback-loops, which allows it to persists the preceding information. See figure 3.7 for an visualisation of an RNN and its feedback-loop. As a result, the RNN also considers the context that went before the current word. Figure 3.8 visualises an RNN, taking previous words (time-steps) in a sentence into account.

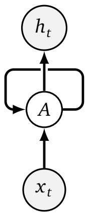


Figure 3.7: RNN visualisation, using a feedback-loop to persist information to the following iteration.

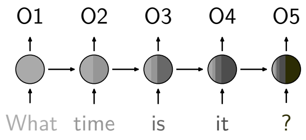


Figure 3.8: Unrolled RNN taking previous words into account.

3.8.1 *Long-Short Term Memory*

Neural Network learn new patterns by updating its weights via back-propagation after a training iteration. In RNN this means it constantly goes through matrix multiplications till it reaches the first step in the time-series. This introduces two main problems for RNNs. First, gradients that are too small will continuously shrink as the back-propagation performs matrix multiplications. In NLP, words which are related might be located far apart in a given sentence. As the RNN back-propagates over a given text, the gradient of the first words become increasingly small. A gradient that is too small prevents a weight from changing its value in any meaningful way, this is called the vanishing gradient problem. Second, a gradient that is too large will update the weights too much, undoing previous training updates. This can even break a ML model as the gradients might explode too much and become NaN.

Hochreiter and Schmidhuber introduced LSTMs in 1997 that solves both problems by introducing the concept of a cell and various gates controlling the state of said cell[HS97; Gre+17]. The gates are different neural networks, which learn whether the information from earlier time steps should be added or removed from the current consideration. Being able to pass relevant information along the chain of given sequences using the gates allows the algorithm to learn which data in a given sequence is important, without having to consider every step in the time-series. The control of these gates also prevents a gradient from vanishing or exploding, as in these cases the gate will not allow the previous information from being considered. See figure 3.9 for a visualisation of an LSTM state and its gates.

[HS97] Hochreiter and Schmidhuber, “Long Short-Term Memory”

[Gre+17] Greff et al., “LSTM: A Search Space Odyssey”

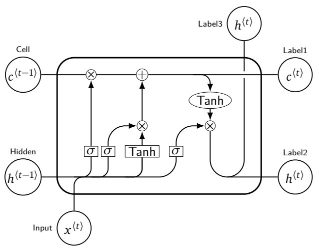


Figure 3.9: LSTM state and gates visualised.

Training an RNN can take a long time, as it considers every step in a sequence separately. Transfer learning helps speed up the training process by using pre-trained weights and refining the weights fast and with fewer data. This method is further explained in chapter §3.9.

3.9 TRANSFER LEARNING

The method of re-using an already developed model as the starting point for a different model and task is called *transfer learning*. In practice, very few people train very large models from scratch with random initialization. It is also relatively rare to have a dataset of sufficient size. Furthermore, training a large model on dozens of gigabytes of textual data is a time consuming tasks, measured in days if not weeks. Instead, it is common to use a model pre-trained on a very large dataset as an initialization layer or input sequence.

An ML model must, as any other application, be thoroughly tested. The methods used to validate and evaluate a ML model are covered in chapter 6.

3.10 REGULAR EXPRESSIONS

Sequences of characters that define a formalized search pattern is called a regular expression. Primarily used to, either extract specific patterns in a string, or replace the matching substring with another string. As a rule based method, it allows for very strict search patterns. Often used in search engines and word processors functionality. An example pattern to match and extract phone numbers is illustrated in figure 3.10

```
(^\+[0-9]2|^\+[0-9]2\(0\)|^\(\+[0-9]2\)\(0\)|^00[0-9]2|^0)([0-9]9$|[0-9\-\s]10$)
```

Figure 3.10: Example regular expression to match Dutch phone numbers.

In this example, the first capture group checks for a country code starting with the character + followed by five numerical denotation alternatives. The second control group checks for groups of numerical sequences separated by punctuation marks and all alternatives. An example output from this regular expression is shown in figure 3.10. Unlike sequence labelling, regular expressions do not alter the content of the string and only extract the matching.

Peter his phone number is <u>+31 06 12345678</u>.

Figure 3.11: Example output of a Dutch phone number regular expression.

Regular expressions can be used to extract entities from text. This thesis uses this approach to create a dataset, more details can be found in chapters §8.2 and §8.3.

4

Related works

▸ SYNOPSIS This chapter analyses related works in NER and using ML to capture unique identifiers. First, listing all the different forms of NER (§4.1). Second, analysing recent advancements in incorporating convolutional DL layers in NLP tasks (§4.3). Third, analysing the results of a two-stage quantized FNN that recognises and classifies named entities separately (§4.4). Fourth, understanding the usage of engineering casing features in an FNN to classify number formats (§4.5).

4.1 TYPES OF NER APPLICATIONS

There are a number of NER subtasks, each specialising in achieving a specific goal. New models and techniques tend to arise when the data does not conform to previously created models, which could be used for transfer learning. Or the content is significantly different or structured in a different format, such as between Wikipedia and Twitter. The outcome of the NER results can also serve very different purposes, such as for reading comprehension or grouping similar documents together.

▸ DOCUMENT CLUSTERING groups similar documents together by related keywords or as an embedding in a shared spatial space[Her+06; Ste+13]. Using the same motivation as a bag-of-words model, if the entities within the documents are similar then the documents are probably similar too. Documents of different languages can be clustered but require individual NER models for each language.

[Her+06] Herranz et al., "Multilingual News Document Clustering: Two Algorithms Based on Cognate Named Entities"

[Ste+13] Steinberger et al., "Multilingual Media Monitoring and Text Analysis - Challenges for Highly Inflected Languages"

▸ READING COMPREHENSION is the process of asking a question in natural language that can be answered by the provided document, mostly used in search engines. Often natural language provides a definition or argument in text. A question to query this answer can be be constructed by prepending words such as *who*, *when* and *why*, followed by a question mark.[1][R.+16].

[1]E. g. the sentence *In meteorology, precipitation is any product of the condensation of atmospheric water vapour that falls under gravity.* contains the question-answer pair *What causes precipitation to fall?* - gravity

[R.+16] R. et al., "SQuAD: 100,000+ Questions for machine comprehension of text"

▸ INFORMATION RETRIEVAL expands on NER by providing an environment where the named entities can be queried upon. NER at its core only tags the named entities it finds, other than that it doesn't do anything. Search engines such as Google or Bing use information retrieval by highlighting the search query as preview in a list of knowledge bases it found. Various systems such as Apache Lucene[2] allow a query of named entities with custom values and properties on a target document[3].

[2]`https://lucene.apache.org/`

[3]An example query could be to: select all documents from where the named entity phone number is equal to +31 (0)612345678.

▸ SUMMARISATION can be applied to list all the named entities in natural language as if its a summary. Often used to get a quick understanding of large documents by mentioning all the key components of the document.

▸ ENTITY LINKING matches an entity to a knowledge base[4], also called disambiguation[HSZ11; McN+11; Hac+13]. Usually by generating candidate entities that might relate by name or value. However, named entities can also be ambiguous such as *cabinet*. Using disambiguation the entity may refer to a *government group* or a *storage closet*.

[4]Wikipedia is a prime example of entity linking. All keywords in a Wikipedia article link to another article.

[HSZ11] Han et al., "Collective Entity Linking in Web Text: A Graph-Based Method"

[McN+11] McNamee et al., "Cross-Language Entity Linking"

[Hac+13] Hachey et al., "Evaluating Entity Linking with Wikipedia"

4.2 RULE BASED AND LEARNING BASED HYBRID APPROACH

Various hybrid approaches and methods have been developed and implemented[RRM06], which automatically discover patterns and create the respective regular expressions to capture said pattern. One could argue that this is a disguised learning based method as it is no longer *hand*crafted and it used an implementation which automatically capture patterns, from which it can learn new patterns by looking at its historical discoveries. Some models incorporate both approaches to capture different types of patterns. Manning et al. use handcrafted regular expressions to capture various numerical sequences (e. g. phone numbers) and machine learning as a learning based method in the STANFORD CORENLP model.

[RRM06] Romano et al., "Automatic Discovery of Regular Expression Patterns Representing Negated Findings in Medical Narrative Reports"

4.3 CONVOLUTIONAL NEURAL NETWORKS IN NER

First introduced in image recognition tasks, CNNs, was introduced and demonstrated as a strong feature. Able to capture both low level and high level features from many data-points, pixels in the case of image recognition. Traditionally, CNNs strength and value was in processing a grid, or matrix, of values. Recently, advancements have been made which demonstrate that CNNs can also be applied to NLP problems.

Documents are transformed to row based records, each row is a separate word. Where the columns are the embedding vector values of that word. The result is a matrix of the same sentence. Using filters the CNN convolves over the matrix, capturing new features which are as rich as n-grams but in a compact representation[Kdn]. Zhang and Wallace designed and developed a CNN architecture for NLP problems. It uses three region sizes: 2, 3, and 4, each which has two filters. Every filter convolves over the sentence matrix and captures new feature maps. Lastly, a 1-max pooling layer is applied on each feature map and concatenated to form a 1-dimensional feature vector. A final layer classifies the feature vector to two categories. The architecture of this model is visualised in figure 4.1. This architecture was published in 2015, since then many advancements in both CNNs and NLP have been made. Zhang and Wallace paper demonstrates that CNNs are capable of capturing patterns from text using convolution. Not to it's full extend, however, as it does not capture words which are related but relatively far apart due to complexity and ambiguity of natural languages.

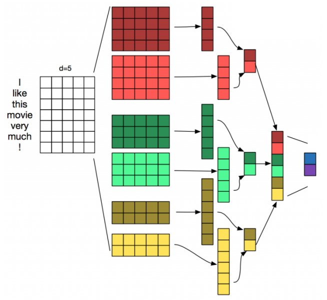


Figure 4.1: CNN model architecture of a sentence, represented as a matrix, convolved into feature maps and concatenated in an activation (classification) layer. Ye Zhang and Byron C. Wallace. "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1510.03820 (2015). arXiv: 1510.03820. URL: http://arxiv.org/abs/1510.03820.

[Kdn], *Deep Learning for Natural Language Processing (NLP) – using RNNs & CNNs*

4.3.1 *End-to-end NER model using concatenated CNN feature maps*

In 2018, Hovy and Eduard published the paper *End-to-end Sequence Labelling via Bi-directional LSTM-CNNs-CRF*. Achieving state-of-the-art results, scoring a $97.55 F_1$ on the CoNLL-2003 test subset [TKSDM03]. The model's architecture consists of a CNN convolving over a matrix representation of each sentence, fed into a bi-directional LSTM, and classified using a Conditional Random Field (CRF) layer. First the words are associated with a POS tag and embedded using GLoVe [PSM14], concatenated as sequence of input vectors. Each word is also convolved, followed by a dropout, max pooling and flatten layer. Additionally the casing for each word is fed as a vectorized encoding. All three feature maps are concatenated and fed to the bi-LSTM, followed by a dense time distributed layer for classification. Hovy and Eduard proved that CNNs and word casings as features have a positive impact on NER tasks.

[TKSDM03] Tjong Kim Sang and De Meulder, "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition"

[PSM14] Pennington et al., "GloVe: Global Vectors for Word Representation"

4.4 QUANTIZED FNN USING NEIGHBOURING WORDS AND BOW FOR CLASSIFICATION

Botha et al. created a quantized Feed-forward Neural Network (FNN) for text classification, with near state-of-the-art performance. The inference model consists of two FNNs. the first scores possible candidate entities, the second classifies the candidate entities to its respective label.

The input text is tokenised, based on space separation. Every token is embedded using character n-grams, with lengths of 1 to 5. Additionally the tokens are hashes and mapped to a fixed set of $20,000$ buckets, reducing the number of stored vectors. The remaining vectors are quantized and reduced to 12 dimensions. See chapter §10.1 for more details on vector and model reduction techniques. All token subsequences of maximum 15 tokens are generated. The casing of each token and a bag-of-words of the input text are created as features[Goo]. The first model generates a score between 0 and 0 for each word, called a candidate entity. Candidate entities which overlap (partially) are removed, selecting the one with the highest score. The second model classifies the candidate entities to its respective label.

[Goo], *The Machine Learning Behind Android Smart Linkify*

During preprocessing, all formatting details of the named entities are removed, such as punctuation marks. And the embedded vectors are reduced astronomically in its dimensions and mapping detail. Therefore, it is remarkable that the model achieves state-of-the-art performance.

4.5 CLASSIFYING NAMED ENTITIES ON FORMAT AND PUNCTUATION

Hayat's manuscript, describes the process of classifying phone numbers using feed-forward neural networks [Hay]. The classification task is divided into four phone classes: international, cellular, nationwide, and local. The data is normalised by removing the first digit of the phone number if it is a zero. As a result, all phone numbers are of equal length, an array of nine digits. Additionally, punctuation marks such as + and - are notated as binary features. The FNN consists of one hidden layer with 25 neurons and an output layer of 4 neurons, equal to the number of output classes.

[Hay] Hayat, *Phone Numbers Classification (PNC) with Feed-forward Neural Networks*

This manuscript, along with previous discussed related works [Bot+17; Goo; HE16], all develop additional features regarding the casings of the token. Each model its accuracy improved significantly with the additional casing features.

[Bot+17] Botha et al., "Natural Language Processing with Small Feed-Forward Networks"

[HE16] Hovy and Eduard, *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*

5

Representing text

▸ SYNOPSIS This chapter opens by explaining why text must be represented by numbers (§5.1). It is followed by detailing two different methods of representing text. First, one-hot encoding and bag-of-words as discrete representation and its limitations (§5.2). Second, using neural networks to assign a vector to embed a word to a spatial embedding space as distributed representation along with its limitations (§5.3). Last, detailing the problem with embedding unique identifier named entities (§5.5).

5.1 OVERVIEW

Natural language is ambiguous on all levels, from phrasing, syntactic to semantic. Texts assume that the reader is aware of the context and understands communication. For example, the word *address* can either mean *to speak to* or *a location*. Without knowledge of the context, the word can mean something very different. And without understanding communication, the intention of the word might be misinterpreted. Therefore, it is crucial for an algorithm to understand the text as a whole, rather than act as a dictionary and know the definitions of words. However, text in its natural form cannot be properly understood by an algorithm due to its logical nature, it must be represented in a different way. There are a number of ways to convert text to a computer understandable format.

5.2 DISCRETE REPRESENTATION

Also called a *sparse* or *local* representation, a discrete representation of text is one method of transforming text to an format that's understandable by an ML algorithm. A simple way to represent a word would be to assign an unique numerical identifier to each word. After all, a computer might not understand natural language but numbers are its speciality. A machine learning algorithm, however, will understand this numerical data to be in some kind of order and map the input to output accordingly. Discrete representation solves this problem by creating a look-up table of the documents vocabulary[1], representing words by its corresponding index position. Often used to represent categorical data for classification algorithms. There are two main discrete representation methods:

[1] A vocabulary is a collection of unique words from the training data.

- One-hot encoding is one of the simplest methods of representing text. It works by distinguishing every word in a vocabulary by a vector of $1 \times N$ matrix. The vector is filled with zeros, except for a single cell with a *one* to identify the respective unique word.

- Bag-of-words improves on one-hot encoding by counting the presence of known words in a document. Measured by comparing the frequency of the words, documents are considered similar if the content is similar. There are two main methods of for scoring the words. Term frequency tallies the amount of times the word appears in the document. And Term frequency - inverse document frequency (TFIDF) is the inverse method of count vectorize, increasing the importance of the rarity of a word. The more a word appears in a document, the less valuable that word is considered in differentiating the given document. Often used in domain specific documents as the used language is more distinct. Both methods are sometimes also called *frequency based embeddings*.

5.2.1 *Limitations of discrete representation*

Context is often required to recognise named entities in documents, even more so the unique identifiers this research aims to recognise. Discrete representations create vectors for words which only contain information about itself. Discarding the relationships between words and not conveying information about the surrounding context. As a result the model would misinterpreted a lot of words, hurting a model its quality and accuracy.

The documents this research concerns itself with are of vastly different types, spanning many different domains. As a consequence, it's nearly impossible to define a single vocabulary for such an expansive and wide range of documents. A discrete representation method cannot represent a word it has not seen before, limiting a model its generalisation and adaptability.

Each word in a document is represented by a vector equal to the size of the vocabulary. And if the vocabulary increases in size, so do the vectors, making matrices more sparse. A very costly method of storing data, increasing the model its memory usage considerably and in turn limiting its performance and scalability. Considering the vast number of documents an implemented model would have to process, this performance limitation is not acceptable.

5.3 DISTRIBUTED REPRESENTATION

Using a set of language modelling and feature learning techniques, words can be embedded as a vector where related words take a similar vector in the spatial embedding space. The main technique to create such vectors is via a trained Neural Network. The main problem of training such NN is the required amount of training data. To get a complete understanding of a word it must be presented in all possible context the word can be found in. The resulting vector is a distributed representation of a word which shares its spatial embedding space with similar words depending on context. The next problem is the number of natural languages. To understand a word among multiple languages it must be presented in all possible context for every different language. BERT is a language model, which extends beyond just embedding documents by also covering multiple languages. It is currently the state-of-the-art language model. Unfortunately, the BERT language model was not used nor tested due to time limitations, it is considered as future works (§15.1).

5.3.1 *Limitations of distributed representation*

Homonyms and homographs are two groups of words that are spelled the same but have different meanings. Embedding these words via a distributed representation results in a single vector that represents two distinct meanings. Various word embedding models such as word2vec use word disambiguation to solve this problem. First identifying the exact word by it's surrounding context before embedding a word in a latent space.

Another limitation of both word representation methods is its indifference to word order and inability to represent idiomatic phrases. For example, the meanings of *France* and *Air* cannot be easily combined to obtain *Air France* [Mik+13].

[Mik+13] Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality"

5.4 CONTINUOUS BAG-OF-WORDS VERSUS SKIP-GRAM CHARACTER EMBEDDINGS

Considering the scaling issues and performance issues of discrete text representations and the higher accuracy of dimensional distributed representations, a number of different (NN) embedding models were tested. A wide variety of different embedding approaches were considered, to determine which model and approach can best embed both words and unique identifiers. The two main considered approaches are: continuous bag-of-words (CBOW) and character skip-gram.

The limitation of word-based models is that it cannot embed words it has not seen before, i. e. unique identifiers cannot be embedded. Instead, a null vector must be initialised so that the token can even be processed and considered. On the other hand, character embeddings can embed any given character sequence, as it assigns weights based on a window of characters, which, combined can form and thus embed any word. Therefore, CBOW embedding approaches were not further considered.Skip-gram has also been proven more accurate. It is trained by predicting masked words in a given sequence, learning a better understanding of the semantic relations between words. Opposed to a CBOW approach, which uses word frequencies in the training data. The character skip-gram embedding model used is called fasttext[Mik+18].

[Mik+18] Mikolov et al., "Advances in Pre-Training Distributed Word Representations"

5.4.1 *Fasttext character skip-gram model*

The number of data samples is limited, as detailed in chapter 8. Therefore, the embedding layer is initialised with one million word vectors trained on the Wikipedia dump dataset, using the fasttext library. The Wikipedia dump dataset was pre-processed using a Pearl script developed by Matt Mahoney[2]. See algorithm 6 in appendix E for the complete script.

[2] http://mattmahoney.net/

The character subsequence lengths of the n-gram was set to one to five. The sequences are mapped to $20,000$ buckets. After pre-training on the Wikipedia dump dataset and refined on the generated (§8), the vectors were quantized, as detailed in chapter §10.1.

5.5 EMBEDDING UNIQUE IDENTIFIERS

The purpose of unique identifiers is to be never-seen-before, the named entities that appear in the training dataset should practically never reappear in production documents. As mentioned before, the main purpose of using a character n-gram is to be able to embed never-seen-before tokens.

To test the added value of embedding unique identifiers, the entities itself, along with its respective related words, were embedded and compared for similarity. First, verbs related to the name of each entity were extracted from the WIKIPEDIA dump and POC dataset trained embedding model using a cosine similarity score of > 0.8[3]. Second, ten entities are generated per named entity using regular expressions. Appendix B lists the used regular expressions. Third, all words and entities are embedded, the resulting 300-dimensional vectors are reduced to two-dimensional vectors using principal component analysis.Four, the first two principal components were visualised on a scatterplot (figure 5.1). The squares depict the embedded words and the triangles depict the embedded entities, the different named entities classes are separated by colour.

The key takeaway was that all entities of all classes share the exact same embedding space (top left cluster). Meaning that the embedding layer cannot distinguish the different unique identifier named entities. This supports the findings and proposed arguments in chapters §4.3 through §4.5. Another important detail is that the words related to the named entities phone number and e-mail address also share the same embedding space. The IBAN named entity, however, is clearly distinguished from the other named entities.

[3] E. g. for the phone number named entity, the final list of words consists of *phone*, *phoning*, *calling*, etc.

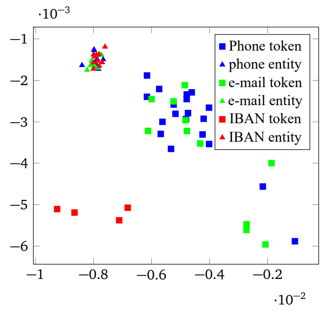


FIGURE 5.1: Embedded unique identifiers and related words on a two-dimensional space.

6

Evaluation metrics

▸ Synopsis This chapter opens by globally explaining how an NER model is evaluated and what the acceptance criteria of the final model should be (§6.1). Three different evaluation metrics are discussed, each improving on the former. First, simple accuracy score and why it is often inaccurate (§6.2). Second, mean squared error and how cross-entropy error improves on it (§6.3). Last, F_1 score as a weighted combination of the precision and recall score (§6.4).

6.1 OVERVIEW

To determine how an NER application performs, a series of evaluation metrics must be defined in order to establish a baseline for acceptable results. Typically the result of NER applications are compared to human annotators on the same dataset. Datasets annotated by a person is called the the *gold standard*. And a dataset annotated by an algorithm is called the *silver standard*. The unique identifier named entities this thesis concern itself with are not annotated in any publicly available dataset. Annotating a dataset by a person, to use it as an evaluation dataset, is too costly. Hence, the NER application will be compared to the already existing solution which the NER application aims to replace, the silver standard. The NER application must out perform the existing implementation to be deemed as acceptable for replacement. These requirements are further detailed in chapter (§14.1.1).

6.2 ACCURACY SCORE

There are various methods to determine a performance score for an NER algorithm. The most common, but crude, method is via an *accuracy* score. Calculating the number of named entities it found and annotated correctly, divided by the total number of samples.

$$\text{accuracy} = \frac{\text{Correctly identified named entities}}{\text{total number of named entities}}$$

6.3 MEAN SQUARED ERROR AND CROSS-ENTROPY ERROR

A more accurate measurement, compared to a simple accuracy score, is the Mean Squared Error (MSE). Suppose a NN is classifying data and the outcome is a range from $0-1$, indicating it's certainty of a correct label classification. To measure such model's accuracy, consider 3 separate inputs for the NN to predict; 2 were predicted correctly, and 1 was wrong. MSE will measure this as a $\frac{1}{3}$ classification error, and $\frac{2}{3}$ correct classification, which results in a 0.67 MSE score. Whereas Cross-Entropy error (CSE) calculates how correct or wrong a given prediction is. E. g. a prediction that was barely correct will receive a lower score in CSE than in MSE. Vice versa, when a prediction was only barely incorrect, MSE sees it as entirely wrong whereas CSE still gives points for being almost right.

6.4 F_1 SCORE

A more accurate and refined evaluation metric is the F_1-*measure*, using a combination of *recall* and *precision* metrics to compute a score. Widely used in the evaluation of NER models[Der16]. Hand and Christen criticise the F_1 score for binary classifications as it considers precision and recall to the same extend, of equal importance. A misclassification cannot be equally attributed to both factors. David Hand et al criticise the use of the F_1 score as it does not take true negatives into account.

[Der16] Derczynski, "Complementarity, F-score, and NLP Evaluation"

This thesis considered using MCC[CJ20] after reading and acknowledging the criticism of the F_1 score by Hand and Christen and David Hand et al's advice but decided to use the F_1 score nonetheless due to the limited available time and the increased complexity of implementing a different evaluation metric. The criticism is mainly directed at binary classifications, which is not the case for this thesis. The implementation of clearer evaluation metrics and also the substantiation[1] of a classification are considered future works.

[CJ20] Chicco and Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation"

[1] E. g. Bayes theorem to communicate clearer the weights and result of a classification.

- **PRECISION** is the number of correctly identified named entities, averaged with the total number of identified named entities.

$$\text{Precision} = \frac{|\,\text{silver} \cap \text{retrieved}\,|}{|\,\text{retrieved}\,|}$$

- **RECALL** is the number of correctly identified named entities divided by the total number of named entities in the dataset.

$$\text{Recall} = \frac{|\,\text{silver} \cap \text{retrieved}\,|}{|\,\text{total}\,|}$$

- **F_1-SCORE** is a scoring measure that considers both the precision and the recall to compute the result accuracy. The F_1 score can be interpreted as a weighted average of the precision and recall, where an F_1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F_1 score are equal[2] [Sci]. The formula for the F_1 score is:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

[2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[Sci], *sklearn.metrics.f1_score*

Part III

AUTOMATICALLY COLLECT AND ANNOTATE NER DATA

7

Binary classification proof-of-concept

► SYNOPSIS This chapter lists the proof-of-concept dataset creation (§7.2) and various ML models based on related work chapters §7.3 through §7.5. The result of each model is analysed to determine if it had to be pursued further.

7.1 OVERVIEW

Before deciding on the final approach of collecting the data and annotating the entities, and the ML architecture to capture the entities, a proof-of-concept implementation was developed. The primary reason was to pre-emptively evaluate the theorised methods and ML models, to assure its quality and prevent time from being wasted. Only phone numbers and e-mail addresses were considered as named entities in this proof-of-concept.

7.2 CREATING DATASET

The positive samples and negative samples were created manually, and expanded and diversified using the Context-free grammar (CFG) module from the PYTHON NLP library *NLTK*[1]. CFG[2] is a set of recursive rules, defined by production rules, to generate strings that describe a natural language. By pre-defining a list of different grammars, an expansive number of different sentences were generated that resemble contexts related to the concerned unique identifier named entities. Thus, with a limited amount of manual work, an expansive and relatively diverse list of different sentences can be generated. However, a thorough

[1] `https://www.nltk.org/`

[2] `https://www.nltk.org/_modules/nltk/grammar.html`

understanding of grammar and a large vocabulary of the respective language is required to cover all possible contexts that an entity can be found in. I. e., generating a complete dataset is too costly (requiring language experts) and work intensive. But for a POC the amount of data and coverage is, to a certain degree, less important. Therefore, CFGs were used in the creation of a POC dataset but not the eventual complete dataset, though the already generated sentences by the POC CFGs were used to additionally supply the eventual dataset of chapter 8. See figure 7.2 for one of the CFGs used to generate sentences about phone numbers and appendix C for all used CFGs. A few examples of generated results from the different CFGs is listed in table 7.1. The sentences do not always make sense to a person but this is also not required, the important detail is that related words are found near the entity. Using regular expressions, random phone numbers and e-mail addresses were generated and added on random positions in the given sentence. The named entities are annotated using the *IO* format, more details can be found in chapter §8.3. The result was a list of sentences that directly mention or refer to the respective named entity, with an equal balance of negative samples and positive samples. Additionally, random noise was added to the dataset by extracting sentences containing a digit or at sign from the CoNLL[SM03] NER dataset using regular expressions. The used regular expressions are listed in appendix B.

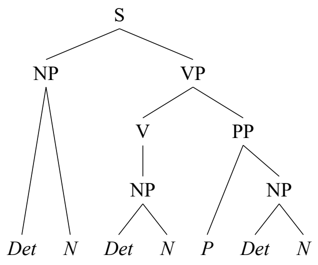


Figure 7.1: Tree visualisation of CFG natural language string generation.

[SM03] Sang and Meulder, "Introduction to the CoNLL-2003 shared task"

```
S → NP VP
NP → Nom | PropN | N
Nom → Adj Nom | N
VP → V Adj | V P [entity] NP | V P [entity] N PP | ...
VP → V Adj | V N
PP → P
PropN → 'him' | 'her' | 'them'
Det → 'the' | 'a'
N → 'person' | 'he' | 'she' | 'them'
Adj → 'quick' | 'long' | 'fast' | 'short'
V → 'calls' | 'called' | 'reaches' | 'phones' | ...
P → 'on'
```

Figure 7.2: CFG rules to generate a sentence mentioning or referring to a phone number.

#	generated CFG string
1	phone number and combination 06-4947384 tried was temporarily online
2	quick he called on 576 847 138 to contact them
3	Tomorrow his authentication code 8371 will expire tomorrow
4	her two factor authentication number 484-393 is valid today
5	quickly he called on 06-4957943 to reach her

Table 7.1: Generated sentences using CFG for POC data.

7.3 LSTM

The purpose of testing an LSTM was predominantly to understand and see the capabilities of finding the named entities by context. First, a given document (sentence in this case) was tokenised, by space separation. Second, two boolean features were engineered, whether the token started with a plus symbol (+) or if a token contained the at sign (@). International phone numbers often start with a plus, and email addresses always contain an at sign. The token sequences (the sentence) was SAME-PADDED to the same length to maintain equal time-steps across the dataset. Each token was embedded using GloVe embeddings, afterwards the casing casing features were concatenated to the embedding to form one input.

Only one version of the LSTM model was tested with default hyperparameters, as it is a proof-of-concept. The precision, recall and F_1 score per named entity can be found in table 7.2. The results were positive and promising, proving that the unique identifier named entities can be found via context. Especially considering that the amount of data was lacking and the named entities were added on random positions in the sentence. An important detail to note is that the classification is binary, which significantly simplifies the task.

	precision	recall	F_1 score
e-mail	0.80	0.64	0.71
phone	0.78	0.76	0.77

Table 7.2: Proof-of-concept LSTM classification report

7.4 FEED-FORWARD NEURAL NETWORK

Botha et al. demonstrated that two-stage FNN models can be as effective as state-of-the-art models (discussed in §4.4 and §4.5). An FNN cannot create new feature maps on its own without DL layers, therefore the token sequences were transformed to a feature-based dataset. First, a given document (sentence in this case) was tokenised, by space separation, this includes the entity itself. Second, the three tokens preceding the first sequence of the named entity and and three tokens proceeding the last sequence of entity are added as separate features. Third, two boolean features are engineered, whether the entire sequence starts with a plus sign (+), and it the entire sequence contains an at sign (@).

The architecture of the ML model architecture is a simple FNN with one hidden layer of 12 neurons. The activation function for the output layer is Sigmoid as the model only outputs two classes. Table 7.3 lists the precision, recall and F_1 score per named entity. The recall is especially good but this is, in hindsight, likely because of the lack of numerical and at-sign based tokens in the data. The results are promising and interesting to investigate further nonetheless, the model is simple, small, and faster than the DL models. This POC model, however, is based on a binary classification and was therefore expected to yield a high result nonetheless, primarily because of the two casing features. It is our expectation that an FNN model with more named entities will perform either similar or worse than the current POC results.

	precision	recall	F_1 score
e-mail	0.72	0.76	0.74
phone	0.65	0.84	0.73

Table 7.3: Proof-of-concept FNN classification report

7.5 SPACY

SpaCy is an open source NLP library [HJ15], written in Python. Providing tokenisers, POS-taggers, embedding strategies, and most importantly a NER model. The exact architecture of the NER model, however, is not published[3]. SpaCy NER uses subword features and *bloom* embedding, and a token based deep CNN with residual connections. The SpaCy library architecture is depicted in figure 7.3.

[HJ15] Honnibal and Johnson, "An Improved Non-monotonic Transition System for Dependency Parsing"

[3]As of writing: 26-05-2020

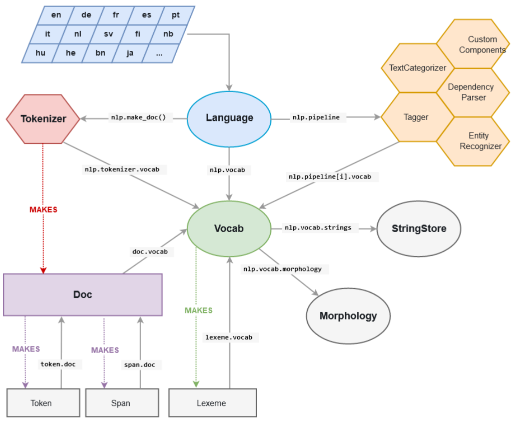


Figure 7.3: SpaCy library architecture.

sources: Dasagrandhi and https://www.youtube.com/watch?v=sqDHBH9IjRU

SpaCy completely pre-processes the given documents itself, therefore there were no additional pre-processing steps except for marking the position of the named entity. For each given document, the named entity must be marked by providing the first character position and the last character position of the entity. As the named entities in the data are manually added (§7.2), the position of each named entity can be calculated.

The model showed optimal performance with 8 epochs, other than that there are practically no hyperparameters to fine-tune. Table 7.4 lists the precision, recall, and F_1-score test results per named entity. Considering that there are only two entities to classify, not counting negatives as *outsides*, it is therefore unexpected to see the low scores. The cause cannot be properly determined because, as mentioned before, the architecture SpaCy its NER model is not published. As a consequence, this model is not further pursued in a full capacity.

	precision	recall	F_1 score
e-mail	0.78	0.48	0.69
phone	0.66	0.62	0.64

Table 7.4: Proof-of-concept SpaCy classification report

7.6 POC CONCLUSION

The POC LSTM model had the best results and proved that the unique identifier named entities can be found via context and a limited amount of feature engineering. The FNN model is feature-wise similar to the LSTM model, using the same casing features and considering context, therefore, it was interesting it showed lower results. The primary difference is that the FNN model has different *hardcoded* features, whereas the LSTM iterates over a sentence on per token basis. Considering the results, creating a custom dataset showed a promising diversity of data.

8

Data and annotations

▸ Synopsis This chapter opens by explaining the purpose of training data for NER ML algorithms, and the reasons why a custom dataset had to be created (§8.1). First, textual data was collected via scraping and the named entities were annotated (§8.2 and §8.3, respectively). Second, combining and preprocessing the data to construct positive samples (§8.4) and negative samples (§8.5) in combination with random noise (§8.6) for a balanced and wide dataset.

8.1 OVERVIEW

Supervised Machine learning algorithms learn to recognise patterns by being fed data with the answers attached. This data will help the model to recognise patterns leading to the correct answer. NER algorithms are trained on a large collection of texts, called a corpus. The named entities within these texts are annotated, teaching the model the linguistic properties and the context surrounding the entity. Most publicly available corpora[SM03; Bal+09; Der+17] with NER annotations only contain wide and common MUC-7[CR98] named entities such as persons, organisations, and locations.

[SM03] Sang and Meulder, "Introduction to the CoNLL-2003 shared task"

[Bal+09] Balasuriya et al., "Named entity recognition in Wikipedia"

[Der+17] Derczynski et al., "Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition"

[CR98] Chinchor and Robinson, "Appendix E: MUC-7 Named Entity Task Definition (version 3.5)"

The named entities this research concerns itself with belong to the ICT domain and are not annotated in any publicly available corpora. The named entities are also inherently unique and can be considered unique identifiers in both textual and numerical form. As a result, the context surrounding the named entities becomes increasingly important. The named entities are usually announced and specified in the direct surrounding context, contained to a single sentence, primarily as the subject or part of the predicate.

Considering the complete lack of available corpora and the inherent unique nature of the entities, a custom collection of texts with annotated entities had to be created. As the context surrounding the entities is always very short, the context length per annotation only needs to be a single sentence long. Thus, a custom collection of sentences containing the named entities had to be created. Ruettet defines a corpus as a representative sample of *actual language production* within a *meaningful context* and with a *general purpose*. Whereas a dataset is a representative sample of a specific *linguistic phenomenon* in a restricted context and with annotations that relate to a *specific research question*. The named entities this research aims to recognize are generally contained within short texts with little context and can be observed as a domain specific linguistic phenomenon. Hence the term dataset is more accurate.

8.2 COLLECTING CONTEXTUAL DATA

[You], *Your Dictionary* `https://www.yourdictionary.com/`

[Worb], *WordHippo!* `https://www.wordhippo.com/`

Various online dictionaries [You] [Worb] provide a *use in sentence* service. Submitting words to this service returns a list of sentences, showcasing how the word can be used in a sentence. Using scraping these sentences were extracted from the online dictionaries. First, the name of the named entity was submitted and extracted. Second, the extracted sentences were POS tagged and all the nouns and verbs were extracted[1]. This list of nouns and verbs related to the named entity was manually curated, removing words that were unrelated to the named entity. The list of curated nouns and verbs were also submitted to the dictionary service, and the result sentences were extracted and added to the dataset. As a result, a dataset of sentences containing words referring to the named entities is created.

[1]E. g. related nouns and verbs to the named entity *phone number* are *smart*)phone and *calling*, respectively.

For a select few entities a dataset or corpus exists which contains the entity naturally in text, but is not annotated. Using regular expressions these entities are found within the texts. The used regular expressions are listed in appendix B. The context surrounding the entities is extracted and added as a sentence to the dataset. The result is a dataset consisting of sentences which, in various forms, mention or refer to the respective entity. See figure 8.1 for a word frequency plot for the phone number named entity. Illustrating that the collected texts are related or similar to the named entity.

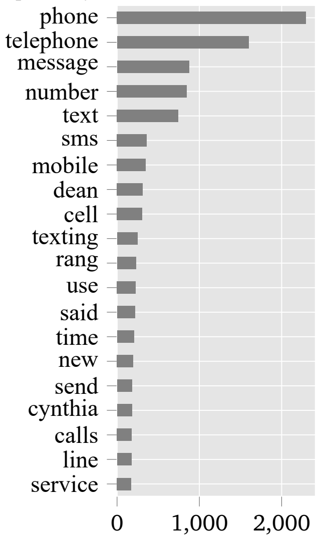


Figure 8.1: Most frequent words in phone number entity related documents.

The scraped data contained unwanted entries and character sequences. To resolve these issues a number of steps were taken to clean the dataset.

▸ DATA CLEANING is the process of preparing data for analysis or training ML algorithms by removing or modifying data that is incorrect, inaccurate, incomplete, irrelevant, or improperly formatted. The scraped data contained unwanted entries and character sequences. Data cleaning was applied to prepare the samples for further pre-processing and eventually as training data. First all empty strings and strings only containing digits were removed. The remainder strings can be considered a complete sentence. Next, for each sentence all occurrences of control characters: *\t*, *\n* and *[Copy]* were removed. Lastly, all non-whitelisted characters were replaced with a space, excess spaces were removed, and all sentences were trimmed for trailing spaces. Algorithm 1 notates the software implementation.

Algorithm 1: Cleaning data.

```
Data: List of strings $S$
Result: List of cleaned strings $d$
1  $d = \emptyset$
2  $g = \{a, b, c, \ldots, z\}$
3  $h = \{\backslash t, \backslash n, \backslash r, [copy]\}$
4  foreach $s \in S$ do
5      $s_i = s_1, s_2, s_3, \ldots, s_n$
6      if $(|s| > 0)$ and $(s_i \subset g)$ then
7          $s$ to lowercase
8          remove $h_i$ from $s_i$
9          trim $s$ white-space
10         append $s$ to $d$
```

8.3 GENERATING AND COLLECTING ENTITIES

The goal of this research is to extract edge-case entities not conforming to its format. The ML algorithm requires an expansive variety of formats per named entity to learn to generalise. If the ML algorithm was only supplied a handful of different formats then the resulting model would be overfit, meaning it will not detect new never-before-seen entity formats. Thus, a list with an expansive variety of formats per named entity had to be created. The creation of these lists consists of two parts.

A. For each named entity a list of patterns is collected. The list is established by consulting and aggregating both in-house and online sources[2] [3]. The list of patterns is used as regular expressions to generate an expansive variety of entities.

B. A select number of entities can be found in publicly available corpora but are not annotated. The entities are extracted using broad and generic regular expressions. Many extracted entities are false positives which are curated and removed by hand. These entities are added to the already established list of the respective entity in part A.

[2]http://www.regexlib.com/

[3]https://regexr.com/

The result is a unique[4] list of entities in many different formats. This list is consulted during the creation of positive samples and negative samples in sections §8.4 and §8.5 respectively.

[4]Meaning that the content of the entity, not just the format, is never the same.

8.4 CREATING POSITIVE SAMPLES

To provide the most relevant context to the named entity it is preferable that the named entity is the subject of the sentence or part of the predicate. All the named entities are nouns which have specific verbs associated only with that noun. If the noun is accompanied by such verb, it is almost guaranteed that entity is the subject of the sentence or part of the predicate[5]. The described positive samples can be recognized and extracted by comparing all the words in the sentence to a list of verbs related to the noun. First, a list of verbs associated with each entity was created by hand. Next, for each verb in the list 50 new verbs are extracted from the embedding model[6] via a cosine similarity score using k-nearest-neighbour (KNN). Lastly, all words of each sentence are compared to this list of verbs. If one of the words matches a verb then the sentence is considered a positive sample. If there were not a substantial enough number of positive samples. Then the positive samples were supplemented by unused leftover sentences, now also considered as positive samples.

[5]E. g. when a sentence contains the noun *phone number* and is accompanied by the verb *calling* then the noun is probably the subject or part of the predicate.

[6]The embedding model is detailed in chapter x. The important part is that all verbs extracted from the embedding model occupy the same spatial position in the embedding space. Meaning the words are similar

All samples that do not meet the minimum word count of seven are supplemented by random non-trivial words such as adverbs, antecedents, adjectives, and unrelated nouns. As a result, the clause is surrounded by randomly picked phrases[7]. These words do not contribute to locating the named entity, teach the algorithm to generalise and learn which words are related to the entity.

[7]By introducing random words to the algorithm it learns to generalise.

So far, all sentences do not yet contain the entities itself. Just words referring to the entity by the subject, name, or predicate. For each sample an entity is extracted or generated as defined in section §8.3. For positive samples the noun of the sentence or a random word is replaced by the entity. Supplemented samples contain fewer references to the named entity and must not be replaced. Here the entity is inserted within one to three word indexes (either left or right) from the the subject, noun, or predicate. However, the entity is never placed within one to two word indexes of the beginning or end of a sentence.

To train an NER algorithm it needs to know the exact position of the named entity in the sentence. Most named entities span multiple separated character sequences. Meaning that the entity is separated by spaces or punctuation marks such as hyphens and slashes. Considering that the named entity can span multiple sequences, a word index cannot be used. Hence, referred to as the entity its span. Rather, the position is indicated by the start character position and last character position of the named entity in the sentence. At this stage, two extra columns were added to the mark the span of the entity. The algorithm uses it to learn the context surrounding the named entity, relating it to similar words. The entity its semantic and syntactic can also be processed and supplied as a feature to the model.

Software implementation

Algorithm 2 details the software implementation. Where s is a string in a list of strings S[8]. K is a list of words related to the named entity, used to scrape the *use-in-sentence* service. V is a list of words related to K extracted from the embedding layer and also contains K itself.

[8]A string is equal to a sentence

Algorithm 2: Pre-processing dataset.

```
Data: List of strings $S$
Result: List of preprocessed positive samples $d$
1  begin
2      $d = \emptyset$
3      $N = \{0, 1, 2, \ldots, 9\}$
4      $A = \{a, b, c, \ldots, z\}$
5      $V$ = generate_action_verbs() $\cup K$
6      foreach sentence $s \in S$ do
7          set $C$ for all characters in $s$
8          if $(|s| > 0)$ and (s contains text) then
9              $s$ to lowercase
10             remove $\{\backslash t, \backslash n, \backslash r, [copy]\}$ from $s$
11             $W = (W_1, \ldots, W_n)$ = split words $W$ in $s$ on space
12             $X$ = generate_entity()
13             $Y = \perp$
14             for $i \leftarrow 0$ to $|W|$ do
15                 if $W_i \in V$ then
16                     $W_i := X$
17                     append string (join $W_n$ by space) to $d$ as
                         positive sample
18                     $Y = \top$
19                     break
20                 if $i = |W|$ and $Y = \perp$ then
21                     $Y = (Y_1, \ldots, Y_n)$ = index position(s) of $W$ in
                         $s$ where $W \in K$
22                     $Y$ = random.sample(($Y$), 1)
23                     insert $X$ at $s_y$
24                     append $s$ to $d$ as supplemented sample
```

8.5 CREATING NEGATIVE SAMPLES

Negative samples teach the algorithm what the named entities do *not* look like, or not in its entirety. The ML algorithm maps a function which output is shared by all the positive samples and by none of the negative samples. The creation of negative samples consist of three distinct parts. All negative samples were created by altering or modifying positive samples. Negative sample are not duplicated from positive samples. If a positive sample is changed to a negative, it is no longer part of the positive samples either.

Part A and B teaches the algorithm to look at both the context and the entity itself. Suppressing matches when the context is conflicting or contradictory to the entity. And part C to correctly recognize and tag the *entire* entity its span. Preventing the model to become a mere unique identifier detector.

A. Words specifically referring to the entity are replaced by words referring to different entities taking similar form. [9] The result is a negative sample with a mixed context referring to a correct entity.

B. The named entity is changed to a different unnamed entity while maintaining the original context. As a result the original correct context now contains and refers to an incorrect unnamed entity. Vice versa of part A.

C. Positive samples are changed to negative samples by changing the span. Either by including surrounding words as part of the span or not capturing the entire entity.

[9] E. g. phrases referring to *phone numbers* are changed to *identification code* or *confirmation number*.

8.6 RANDOM NOISE SAMPLES

The number of samples in the dataset is limited. Especially after various positive samples were required to be re-purposed to negative samples for quality control. Small datasets can introduce problems when training ML algorithms. The first and foremost problem is that the model might learn the specific input samples and associated output instead of learning a general mapping of inputs to outputs. Meaning the ML algorithm is less able to recognize training samples as it is changing constantly, resulting in smaller weights with a lower generalization error[GBC16]. Thus, to explicitly teach the ML algorithm to generalise, random noise is introduced as data samples.

[GBC16] Goodfellow et al., *Deep Learning*

Random sentences were extracted from the *enwiki dump* dataset, a collection of Wikipedia articles. The same cleaning and preprocessing steps were applied. The sentences are completely random from an expansive range of different domains with no special characters or HTML tags. For each sentence an *unnamed* entity taking a similar but different form to a named entity is inserted.

The second problem is that a small dataset provides fewer samples to describe the input space and its relationship to the output. Simply upscaling or duplicating samples with the same context but a different entity, and vice versa, would work counter-intuitive as it will not bring variety to the input space and is prone to overfitting. Wei et al. created and defined four operations in to increase data variety, preventing overfitting, and create a robust model[Wei+19].

[Wei+19] Wei et al., *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*

1. **Synonym replacement**: Replacing words or phrases with their synonyms is a proven way to generate data to improve generalisation without compromising quality[Zha+16]. Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.

[Zha+16] Zhang et al., *Character-level Convolutional Networks for Text Classification*

2. **Random insertion**: Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this n times.

3. **Random swap**: Randomly choose two words in the sentence and swap their position for n times. Preventing the model of learning the same input construction.

4. **Random deletion**: Randomly remove each word in the sentence with probability p. Reducing the chance that the same key elements occur in too many samples.

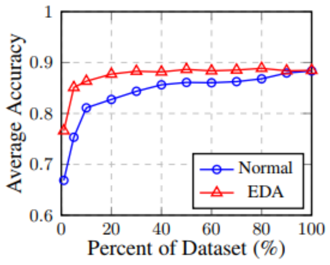


FIGURE 8.2: Performance on text classification tasks with respect to percent of dataset used for training. Using data augmentation operations significantly outperforms on normal training on small datasets. Zhang et al. *Character-level Convolutional Networks for Text Classification*. 2016. URL: https://arxiv.org/abs/1509.01626

These data augmentation techniques were applied on the dataset. The result might be nonsensical to humans. This, however, is exactly the random noise required to make a model more robust. [Zha+16] proved that data augmentation on NLP datasets outperforms regular training performance at almost all dataset sizes. Figure 8.2 shows that a dataset using only 50% of the available samples using data augmentation achieves the same accuracy as the normal samples using all available data. Especially smaller datasets improve on quality and thus model performance.

9

Feature engineering

▸ SYNOPSIS This chapter explains the purpose of adding additional features to an NLP dataset (§9.1), detailing the features and its implementation on a per chapter basis (§9.2 trough §9.6). Last, concluding the resulting dataset and providing a list of all engineered features (§9.7).

9.1 PURPOSE OF ADDITIONAL FEATURES

A feature in machine learning can be defined as an individual measurable property or characteristic of a phenomenon being observed [Bis92]. Features describe the input data with more details to help an ML algorithm distinguish the data and substantiate a classification or prediction. In a more classical linear algebraic ML problem there is often an expansive variety of possibilities for feature engineering[1]. In textual data, however, there are often fewer options for feature engineering. The base primary features of textual data are the words itself. These words can be described by engineered features, the purpose of an additional token based features is to highlight its different properties to an ML algorithm[2].

[Bis92] Bishop, "Pattern recognition and machine learning"

[1] E. g. of any incremental or numerical value the maximum, minimum, mean, and medium of the range can be calculated.

[2] I. e. a name always starts with an capital, which can be turned into a binary feature for any token. And an e-mail address always contains an at-sign (@), which again can be described as a binary feature: does the token contain an at-sign or not?

The features that are engineered to enhance the current dataset quality are further discussed and detailed in chapters §9.2 trough §9.6.

9.2 SENTENCE TOKENISATION

The dataset format thus far is a collection of sentences with the location of the named entity specified. A ML algorithm cannot be fed the document as a single large character sequence, otherwise it would consider the entire document as one object in its entirety. The document must be separated into its logical pieces, sub sequences. The process of separating a character sequence into a defined document pieces is called tokenisation, where each piece is a token.

Considering that texts are sequential, the sentences are separated on a word and punctuation basis. A simple method of tokenisation is to split a text on spaces. However, this does not suffice as it will not separate punctuation marks, which are attached to words. The Python library, NLTK[3], provides a submodule called *nltk.tokenize* which does separate words and punctuation marks into different tokens.

[3] https://www.nltk.org/

Each token is added as a new record in the dataset, a *sentence ID* keeps track of which word belongs to which sentence. The order of words is preserved and does not require an *order ID* column. As a result, the data is changed to a time-stepped format. Where the time factor is the order of words in a document. Especially RNNs models benefit from this format.

For each token record, the two preceding tokens, and the two proceeding tokens are added as features. Providing the token with its direct surrounding context as separate features. If the first preceding token and the second preceding token does not exists, because the current token record is at the start of a sentence, then the values are set as __*START2*__ and __*START1*__ respectively. Vice versa for the proceeding tokens of the current token. If the first proceeding token or the second proceeding token does not exists, because the current token is at the end of a sentence, then the values are set as __*END1*__ and __*END2*__ respectively. These additional tokens features are primarily used in FNN models since this architecture considers one data entry as a whole. The token features will henceforth be refered to as the *token-at-position* features.

9.3 TOKEN CASING AND SPECIAL CHARACTERS

For every token feature, per dataset record, the casing of said token is registered. The additional casing feature helps a ML model recognise various specific named entities. E. g. a phone number will always be either completely numerical or numerical with punctuation marks as dividers, and an e-mail address always contains the @ symbol. The following list of casings per token are captured:

- is a digit
- contains a digit
- is primarily numerical
- contains @

- is lowercase
- is uppercase

Using a list means that multiple casings per token are registered. The list is one-hot encoded to make sure that the ML algorithm interprets the feature as categorical. The software implementation is detailed in algorithm 3.

Algorithm 3: Registering the casing features of a token.

```
Data: token $t$
Result: Encoded list of $t$ casings
begin
    $c = \emptyset$
    if $t$ is digit then
        c add {is digit}
        break
    else if $t$ contains digit then
        $X = 0$ for $C \in t$ do
            if $t$ is digit then
                $X += 1$
        $X = \frac{X}{len(t)}$
        if $X > 0.5$ then
            c add {primarily numeric}
        else if $t$ is numerical then
            c add {numerical}
        else
            c add {contains digit}
    if contains @ then
        c add {contains @}
    if contains punctuation then
        c add {contains punctuation}
    if is upper then
        cing add {is upper}
    else if is lower then
        c add {is lower}
    vector encode c
    return encoded c
```

9.4 STEMMING

Words can change grammatically depending on the context, while often conveying the same meaning, such as *organize*, *organizes*, and *organizing*. This applies to words with similar derivational meanings, such as *democracy*, *democratic*, and *democratization*[Sta]. An embedding layer creates different vectors for each word, giving different weights to, what's in practice, the same meaning. Stemming a word involves cutting it to its root form. As a result, the words would have the same embedding vector and thus weighed the same. Often also used as a form of dimension reduction or normalisation. Cutting down on the value size of features for the sake of reducing storage size and computational time.

[Sta] Stanford, *stemming and lemmatization*

The problem is that affixes can also create or expand new forms of the same word, or even create new words themselves[4]. In English, prefixes are always derivational, but suffixes can be both derivational[5] and inflectional[6] [Rab19].

[4]Adding the affix *-eco* in front of *system* creates a completely different word *ecosystem*.

[5]Adding the suffix *-ist* to *guitar* creates a new word *guitarist*, which is very different from *guitar* while only three letters were added.

[6]The affix *-er* in the word *faster* makes it significantly different from its root form *fast*.

[Rab19] Rabby, *What is NLP & What Do NLP Scientists Do?*

There are various algorithmic approaches which take all the aforementioned problems into consideration and returns the most optimal stem depending on the word and its context. This thesis uses the Snowball stemming algorithm [Por][7]. Applied on all tokens to create additional features. The tokens in the dataset remain *as found*. Extra columns are added for the stem version of those tokens. An example output of stemmed tokens is illustrated in figure 9.4. One important detail is that stemming is not intended to always return a grammatically correct word, such as detailed in the example figure. All tokens per dataset record are stemmed and added as additional features in new columns, the tokens remain unaltered.

[Por] Porter, *A language for stemming algorithms*

[7]https://snowballstem.org/

The	phone	numbers	of	the	employees	are	in	the	registry
the	*phone*	*number*	*of*	*the*	*employ*	*are*	*in*	*the*	*registri*

Figure 9.1: Stemmed text example output.

9.5 LEMMATISATION

A lemma is the dictionary form of a word, lemmatisation is the process of determining the lemma of a word and it's meaning in context. Often by grouping the various forms of a word as a single item. Similar to stemming, lemmatisation is used to group various forms of words together. To create a single embedding vector for the same word with the same meaning. So that it's weighed the same in a ML model. It differs, however, as the words remain grammatically *intact* and keep more context value. An example of lemmatised words is illustrated in figure 9.2.

Supposedly	he	was	calling	while	studying	for	his	exams
suppose	*he*	*was*	*call*	*while*	*study*	*for*	*his*	*exam*

Figure 9.2: Lemmatised text example output.

Stemming uses a dictionary lookup to determine a word it's dictionary form. WordNet is a large and publicly available English lexical knowledge base[Mil+90][Wora]. The Python library, NLTK[8], provides an interface to WordNet, to be used as lookup dictionary for lemmatisation. Additionally, the POS tag of the word is supplied, which helps the lemmatisation accuracy by understanding the position of the word.

[Mil+90] Miller et al., "Introduction to WordNet: An On-line Lexical Database"

[Wora], *What is WordNet?*

[8]https://www.nltk.org/

All token-at-position features per dataset record, the current token, two preceding tokens, and the two proceeding tokens are lemmatised and added as additional features in new columns, the words remain unaltered.

9.6 PART-OF-SPEECH

Every word in a sentence can be divided into categories based on its definition and context via word disambiguation and the word its grammatical properties, such as tense, casing, plural or singular, etc. A Part-of-speech (POS) tag adds more understanding of the context to a ML model, often used in corpus searches and text analysis tools[9].

[9]E. g. in the sentence "give me your answer", *answer* is a noun, but in the sentence "answer the question", *answer* is the verb.

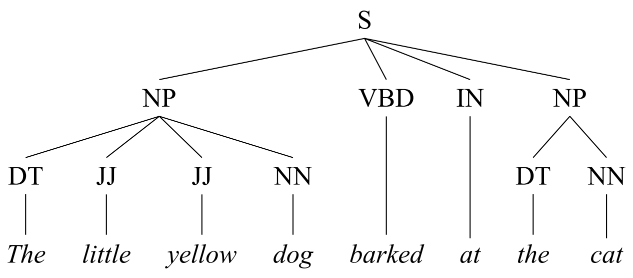


Figure 9.3: Tree visualisation of POS tagged text.

There are multiple different techniques for part-of-speech tagging. Lexical based methods assign the POS tag to the most frequently occurring words in the data. Rule based methods use static grammatical properties to categorizes words[10]. Often these two methods are combined to tag words that aren't present during the training of the data[Ram18].

[10]E. g. words ending with *-ed* or *-ing* are assigned as verbs.

[Ram18] Ramachandran, *NLP Guide: Identifying Part of Speech Tags using Conditional Random Fields*

Probabilistic methods categorize words on the probability of a particular tag sequence occurring.

This thesis uses a pre-trained probabilistic approach, from the library NLTK[11], to label POS tags to token. Table 9.1 lists all POS tags that NLTK can assign to a token. The library returns a tuple, containing the original token and its POS tag. Figure 9.4 shows an example POS output. The POS tag for each token in the dataset is added as feature. As a result, each data record contains five POS tags: the original token, the two preceding tokens, and the two sequential tokens.

[11] https://www.nltk.org/

Don't	forget	to	give	mister	Peter	my	phone	number	.
VBP	*RB*	*VB*	*TO*	*VB*	*NN*	*PRP$*	*NN*	*NN*	.

FIGURE 9.4: Part-of-speech tagged text example output.

Tag	Description	Tag	Description
CC	coordinating conjunction	PRP$	possessive pronoun
CD	cardinal digit	RB	adverb
DT	determiner	RBR	adverb, comparative
EX	existential	RBS	adverb, superlative
FW	foreign word	RP	particle give up
IN	preposition	TO	to
JJ	adjective	UH	interjection
JJR	adjective, comparative	VB	verb, base form
JJS	adjective, superlative	VBD	verb, past tense
LS	list marker	VBG	verb, gerund
MD	modal	VBN	verb, past participle
NN	noun, singular	VBP	verb, sing. present
NNS	noun plural	VBZ	verb, 3rd person
NNP	proper noun, singular	WDT	wh-determiner
NNPS	proper noun, plural	WP	wh-pronoun
PDT	predeterminer	WP$	possessive WP
POS	possessive ending	WRB	wh-adverb
PRP	personal pronoun		

TABLE 9.1: All possible POS tags assigned by NLTK.

9.7 FEATURED DATASET STRUCTURE

After feature engineering the dataset was transformed to a time-stepped format. The direct context surrounding the token is added and every token is stemmed, lemmatised. Additionally, the casing for each token is registered as a categorical one-hot encoded feature. See table 9.2 for an overview of all resulting features.

prev-prev-word	lemma
prev-prev-pos	stem
prev-prev-lemma	casing
prev-prev-stem	next-word
prev-prev-casing	next-pos
prev-word	next-lemma
prev-pos	next-stem
prev-lemma	next-casing
prev-stem	next-next-word
prev-casing	next-next-pos
word	next-next-lemma
label	next-next-stem
pos	next-next-casing

Table 9.2: List of all features after feature engineering the dataset. Ordered from preceding, to current, to proceeding token(s) features.

Part IV

RECOGNISE AND CLASSIFY UNIQUE IDENTIFIER NAMED ENTITIES

10

Reducing word embedding and model sizes

► Synopsis This chapter explains different vector reduction techniques (§10.1), while maintaining embedding accuracy. And bucketing methods (10.2) to reduce padding lengths in training batches. Last, detailing the software implementation in chapter 10.2.

10.1 REDUCING VECTOR SIZE

As ML algorithms train on increasingly larger datasets, especially in DL, the resulting ML models are also increasing in size. In NLP this is especially the case for the embedding layers. Often creating 300-dimensional vectors for every unique word that appeared in all documents combined. For example, the embedding model BERT-base has 100 million parameters. The bigger variant, BERT-large has 340 million parameters. Consequently, size reflects the required computing resources. BERT is trained on GPU'S with 16GB VRAM and fine-tuned on a cloud TPU with a 64GB RAM. At the time of writing, the average desktop computer *only* has 8GB RAM to 16GB RAM and 3GB VRAM to 6GB VRAM. It's needless to say that these models are ridiculously large and increasingly difficult to deploy, no matter the available resources and infrastructure budget.

To reduce ML model sizes, this thesis, compressed word embeddings via product quantization[And16]. Followed by storing the compressed word embeddings in buckets using Inverted File Index, as pioneered by Jégou et al. Using the Encoding Residuals technique to maintain vector accuracy between buckets during compression. All models

[And16] Andrews, "Compressing Word Embeddings" proposed in this thesis were tested with and without model reduction techniques applied, indicated per result table.

▸ Product Quantization can reduce a vector its size greatly while maintaining a significant portion of its performance. Not only does it reduce a model its storage size, but also its memory usage and increasing its memory access times for training times and classifying a large number of documents[Zaf+19]. Dimension reduction methods such as Principal Component Analysis (PCA) can also reduce a vector in size, but will also lower its performance. Because the vector is a representation of a word or context, instead of being a solely numerical value.

[Zaf+19] Zafrir et al., "Q8BERT: Quantized 8Bit BERT"

A simple, but seemingly crude method, to compress embedding vectors is by storing less precision. Product Quantization compresses the vector by dividing its large set of points into groups of approximately the same similarity[Jou+16]. The vector is divided into n sub-vectors, the result is a matrix. Where the number of columns is equal to the number of n divided sub-vectors, and the rows still represent the complete word. See figure 10.1 for a simple visualisation. K-means clustering is applied on each sub-vector, where $k = x$. For each n sub-vectors there is now a set of x centroids, the mean of the cluster. Representing the most common patterns within the sub-vector cluster. Each n sub-vector is replaced by its respective x centroid[Chr]. This reduces a vector its accuracy regarding a specific word but, as results will show, not in any significant or accuracy related impactful way. Figure 10.3a depicts an example of a sub-vector its centroid.

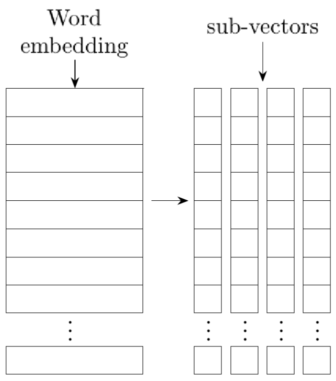


Figure 10.1: Dividing word embedding into n sub-vectors.

[Jou+16] Joulin et al., "FastText.zip: Compressing text classification models"

[Chr], *Product Quantizers for k-NN*

▸ Inverted File Index maps all words (or larger texts) to a position, an index. Clustering the dataset into a number of partitions, called *buckets*. As a result, each vector belongs to a single bucket. To find a word its respective vector, an embedding algorithm only has to look in one bucket. See figure 10.2 for a visualisation.

One problem is that some similar *nearest neighbours* vectors can be cut-off into different buckets. The simple solution is to check KNN buckets for similar vectors as well. Searching vectors in multiple buckets is a performance loss, but still a huge performance gain compared to checking the entire embedding space[Chr].

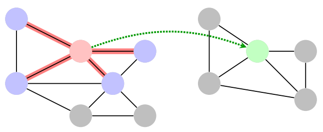
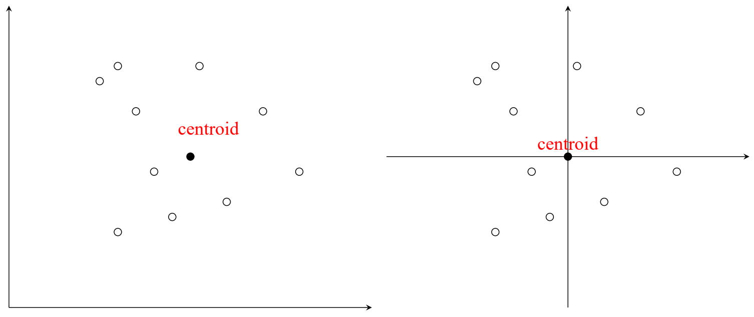


Figure 10.2: KNN searching multiple buckets

▸ Encoding Residuals adds information about the IVF step into the product quantizer to improve vector accuracy. A residual is the offset from the centroid. By replacing a vector with its residual the cluster of "*vectors*" is centered around 0. Figure 10.3b depicts the centroid of the encoded residuals. The nearest neighbour of a vector can now be calculated by the vector its residuals. As a result, the L2[1] distance between vectors from different buckets can be calculated to query the nearest neighbour, so long as all the vectors are replaced by the residuals[Chr].

[1]L2 is the Euclidean distance. The shortest distance between two points in an N dimensional space (the embedding space).



(a) Centroid of an embedding sub-vectors. (b) Centroid of sub-vectors residuals.

Figure 10.3: Centroid and encoded residual visualisation.

10.2 SEQUENCE PADDING REDUCTION VIA BUCKETED SEQUENCES

Sequences are not padded to the same length based on the whole data, but on a batch level. Meaning that the sequences are padded and truncated on a batch level in a data generator, so that the length of the sentence in a batch can vary in size. Additionally this can be further improved by not truncating based on the length of the longest sequence in batch, but based on the 95% percentile of the lengths within the sequence. Improving runtime significantly, while maintaining robust training data. By reducing training time, consequently, more time could be spend on refining the models. See figure 10.4 for a sequence length bucketing visualisation and figure 10.5 for an example.

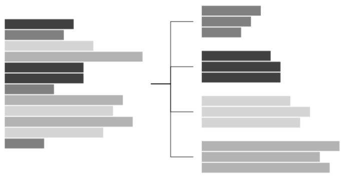


Figure 10.4: Sorting sequences visualised.

	0	1	2	3	4	5	6	7
Bucket i	My	phone	nr	is	548	847	PAD	PAD
	The	site	lists	the	cell	phone	06	43375

	0	1	2	3	4
Bucket j	Call	me	today	on	548 847
	546 645	is	my	cell	PAD

Bucket k ...

Figure 10.5: Padded sequence bucketing

Software implementation

Deep Learning models can work with multiple inputs of different variable shapes. However, a fixed input length in a DL model can improve performance noticeably, especially during training. A fixed input length allows for the creation of fixed shaped tensors and therefore more stable weights.

Algorithm 4 details the implementation of reducing vector sizes using product quantization and using encoded residuals to improve accuracy. The documents are separated into a fixed size number of buckets by the length of the document. This allows the models to train on batches with as few padding as necessary. This algorithm is based on Tbennun repository [Tbe18].

[Tbe18] Tbennun, *tbennun/keras-bucketed-sequence*

Algorithm 4: Sorting padded sequences into buckets.

```
Data: List of sequences $S$
Result: Bucketed padded sequences
Function bucketing_sequences($num\_buckets, batch\_size, seq\_lengths, S$):
    $bucket\_sizes$, $bucket\_ranges$ = np.histogram($seq\_lengths$, bins=$num\_buckets$)
    shape = $(1, )$ if $len(S.shape)$ then shape = 2
    else $S$.shape[2:]
    $actual\_buckets = []$
    for _, $bs$ in $enumerate(len(bucket\_sizes))$ do
        if $bs > 0$ then $actual\_buckets$.append($bucket\_ranges[i+1]$)
    $actual\_buckets\_sizes = []$
    for $bs$ in $len(bucket\_sizes)$ do
        if $bs > 0$ then $actual\_buckets\_sizes$.append($bs$)
    $bucket\_seq\_len = []$
    for $bs$ in $len(bucket\_sizes)$ do
        $bucket\_seq\_len$.append(int(math.ceil($bs$)))
    $num\_actual$ = len($actual\_buckets$)
    $bins = []$
    for $bsl$, $bs$ in $zip(bucket\_seqlen, actual\_bucketsizes)$ do
        $bins$.append(np.ndarray($[bs, bsl]$ + list($input\_shape$), dtype=$x\_seq$.dtype))
    $bctr = [0] \times num\_actual$
    for $i, sl$ in $enumerate(seq\_lengths)$ do
        for $j$ in $range(num\_actual)$ do
            $bsl = bucket\_seq\_len[j]$
            if $sl < bsl$ or $j == num\_actual - 1$ then
                $bins[j][bctr[j], : bsl] = x_seq[i, -bsl :]$
                $bctr[j] += 1$
                break
    $num\_samples = x\_seq$.shape[0]
    $dataset\_len = 0$
    for $bs$ in $actual\_bucketsizes$ do
        $dataset\_len +=$ math.ceil($bs/batch\_size$)
    return ...
```

11

Bi-directional Long-Short Term Memory (Bi-LSTM)

▸ Synopsis This chapter describes the bi-LSTM-CNN-CRF model in detail and elaborates on the benefits of the base LSTM layer (§11.1) and features (§11.2). Each consecutive section introduces additional deep learning layers and features to the model its architecture that improve its accuracy (§11.3 through §11.5). Last, the results are concluded and analysed (§11.6).

11.1 OVERVIEW

For NLP problems it is often beneficial to consider both the past and future of a given context. LSTMs, however, only considers information in its hidden state from the past. As related work, Kiperwasser and Goldberg, and Hovy and Eduard, demonstrated a very effective solution, a bi-directional LSTM. The documents are presented forwards and backwards to two separate hidden states, capturing both past and future information respectively. Both hidden states are concatenated and presented as new feature map to a following layer[HE16]. See figure 11.1 for a visualisation of a bi-directional LSTM. This algorithm is the core of the architecture for this model, each following chapter improves on the former by changing or adding new features and layers, and concluding the results in chapter §11.6.

[HE16] Hovy and Eduard, *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*

The LSTM model trained using the TRAIN ON BATCH function has a performance increase over the FIT and FIT GENERATOR functions when training on a large dataset, which is not easily serialisable. More importantly, the TRAIN ON BATCH function allows for greater control over the LSTM state, and the usage of sequence bucketing, further described in chapter §10.2. It allows the use of batches with varying sizes while the LSTM considers every data sample to the same extend, using state resets. Simply using the fit function would train the model on all data series without resetting the LSTM hidden states.

The used loss function is the sparse categorical cross-entropy. It is the same as the categorical cross-entropy loss function, however, the function accepts an integer target label instead of an one-hot encoded vector. The sparse method was used as it simplified the input data label preprocessing. The Categorical Cross Entropy loss function is actually a combination of a Softmax activation and a cross-entropy loss method. It allows for multi-classifications, returning a list of most likely classifications, granting the results interpreter more workable information.

For any RNN ML algorithm it is more effective to use pre-padding, rather than post padding. An RNN remembers fewer preceding data samples, as the RNN iterates and progresses over the input data. This architecture, however, uses a bi-directional LSTM, which considers both directions of a given input sequence. Hence, padding is evenly applied on both sides, to prevent a bias occurring in one of two LSTMs.

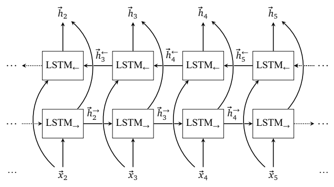


Figure 11.1: Bi-directional LSTM hidden state cells visualisation.

Each version of this LSTM based model in the consecutive sections were re-trained and tested after applying a hyperparameter grid search. I. e. the results of the models listed per chapter are based on the final, most optimal, version of said feature and layer architecture configuratiown.

11.2 RECURRENT TOKENS AND ENGINEERED FEATURES

The input sequences are sorted by its number of tokens and grouped in buckets. Per bucket, the sequences are padded to the same maximum length, preventing the entire dataset to be padded to the sequence with the most amount of tokens, and reducing the number of padding tokens for all other sequences and significantly speeding up the ML model training time. This process is further described in chapter §10.2 along with the detailed software implementation. Setting the LSTM hyperparameter time-step to the padded sequence length allows the bi-directional LSTM to consider the entire sequence from both directions[1]. Additionally, the casings of the token is extracted and encoded to a vector form, as detailed in chapter §9.3. Using an encoded vector form allows the algorithm to consider it as a categorical feature.

[1] I. e. the LSTM algorithm considers one input token at a time but retains information from its previous iterations. It is important to remember that the LSTM network only considers one token and its features at a time.

The model was trained with the following hyperparameter configuration: 15 epochs, 50 hidden LSTM nodes, SOFTMAX output activation function, and NADAM as optimiser, as determined from the grid-search results in section §11.6. Table 11.1 lists the models precision, recall, and F_1-score test results per named entity.

	precision	recall	F_1-score	support
phone	0.58	0.67	0.62	476
e-mail	0.51	0.69	0.59	461
IBAN	0.57	0.86	0.69	182
pers. ID	0.89	0.55	0.69	96

TABLE 11.1: bi-LSTM with engineered features model results.

The added casing features show an improvement over the POC results, when taking the extra classes into account over the POC binary classes.

11.3 CONVOLVING INDIVIDUAL EMBEDDED TOKENS (CNN)

An important aspect and key feature of unique identifiers is the (often) established and repetitive pattern. This is partially introduced to the ML with the token casing features. Telling the model whether the token is capitalized, a digit, or contains punctuation. This, however, is rather binary and shallow. Take for example phone numbers, often written with hyphens (-) to improve readability, separating the number into even groups of digits. The binary casing feature of whether the token contains a punctuation mark does improve a model's accuracy. A convolution across the matrix of characters in that phone number, however, takes all the punctuation marks into account and transforms it to several feature maps. The CNN uses a 3 × 3 filter, the purpose is to capture special character or punctuation marks if it is surrounded by other characters. In the case of e-mail addresses and personal identifier, respectively, the defining recognisable format is the *at* sign that is always surrounded by alphabetical characters, and periods surrounded by numericals, at all times. See figure 11.2 for a visualisation of a 3 × 3 filter that is concatenated to the original input, where x is the number of embedding dimension and k the feature map.

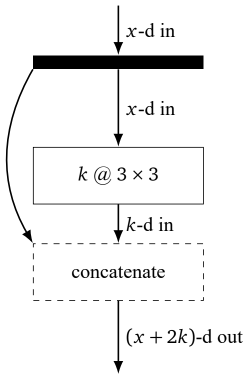


FIGURE 11.2: Convolutional filter K of size 3 × 3, concatenated with the original input.

The model was trained with the same hyperparameter configuration: 15 epochs, 50 hidden LSTM nodes, SOFTMAX output activation function, and NADAM as optimiser. The convolution layer is configured with: 10

filters, windows size of 3 × 3, with a stride of 1. Table 11.2 lists the models precision, recall, and F_1-score test results per named entity.

	precision	recall	F_1-score	support
phone	0.84	0.97	0.90	471
e-mail	0.96	0.96	0.96	455
IBAN	0.89	0.91	0.90	182
pers. ID	0.97	0.60	0.74	96

Table 11.2: bi-LSTM with CNN layers on character-level results.

The CNN added an enormous improvement over the previous model version. The theory is that the CNN filters create feature maps based on the format of the named entity, especially engineering features based on the separation of entity segments, the recurrence of characters, and learning patterns constraints[2]. Due to time constraints the reason why CNNs added such a performance increase was not researched, one such method could be to analyse the activations of the different feature maps.

[2]E. g. the *at*-sign (@) in an e-mail address is always surrounded by regular alphanumerical characters.

11.4 CONVOLVING SEQUENCES OF TOKEN EMBEDDINGS (CNN)

In many natural languages words which are closely related can still be located far apart from one another in a given sentence. Consider the following sentence: ”*Peter phoned the new number of Michael, which is 06 4257453.*”. The theory was that convolution filter with window size 4x4, along with a max pooling layer, would capture the words *phoned* and *number* can create a stronger combined feature as input to the ML model. The feature map is only slightly diminished in value by the noise tokens *the* and *new*, this example does not take the removal of stop-words into account.

The model was trained with the same hyperparameter configurations as the previous LSTM-CNN model. The only difference is that the CNN convolves over all the tokens in a given document as a matrix, instead of per individual tokens. Table 11.3 lists the models precision, recall, and F_1-score test results per named entity.

Adding a CNN based on a matrix of tokens, rather than a matrix of characters (§11.3), along with the casing features (§11.2) showed only limited, or even marginal, improvements in a few named entities. As proven, using contextual features to recognise and classify unique identifier named entities does improve the results, however, adding a CNN barely adds any value that a bi-LSTM had not yet considered.

	precision	recall	F_1-score	support
phone	0.53	0.69	0.60	471
e-mail	0.48	0.66	0.55	451
IBAN	0.59	0.72	0.65	195
pers. ID	0.96	0.56	0.69	96

TABLE 11.3: bi-LSTM with CNN layers on token-level results.

11.5 CRF OUTPUT LAYER CONSTRAINTS (CRF)

Conditional Random Field (CRF) adds constraints to the predicted labels to add assurance to its validity, these constraints are automatically learned by the CRF layer during the trainings process[Cre]. The previous bi-LSTM-CNN model (§11.3) is accurate in recognising and classifying the named entities, as can be observed from table 11.2, but the assigned annotation is often not entirely correct. The bi-LSTM-CNN model tagged the first sequence of an entity with an *I-* instead of *B-*, or a single token long entity with *B-* instead of *U-*[3]. The CRF layer increases the accuracy of the recognised span of a named entity and the constraints, such as the first token of a named entity is classified as *B-* for beginning.

[Cre], *CRF Layer on the Top of BiLSTM - 1*

[3]I. e. the model does not necessarily perform better in terms of classification, but is more accurate in determining the position of the label of a given sequence.

The bi-LSTM-CNN model remains the same as detailed in chapter §11.4, the CRF layer is appended to the bi-LSTM. The input of the CRF layer is the output of the LSTM layer, which is a score for each classification label, without requiring any intermediate steps. The output activation function had to be changed to CRF_LOSS from the KERAS_CONTRIB PYTHON library, all other hyperparameters remained the same. As the results are shown on a named entity level(phone, e-mail, etc) and not on a annotation level (BILOU tagging), there is no meaningful difference from the previous model. Hence, no model testing results are shown for this architecture.

11.6 BI-LSTM-CNN-CRF NAMED ENTITY RECOGNITION MODEL

The final architecture of the LSTM approach is a bi-LSTM-CNN-CRF model. To test the optimal hyperparameters values of this model architecture a custom grid-search was implemented, similar to SKLEARN GRIDSEARCH module. Not every hyperparameter could be cross-tested as the total number of different ML configurations would be $72,900$. Instead, each hyperparameter was tested on the base model. If a different

hyperparameter	value
Epochs	5
optimizer	Nadam
Learning rate	0.01
Output activation func	softmax
LSTM hidden nodes	200
CNN kernel size	4
CNN nr of filters	30
CNN stride size	1
CNN activation func	tanh
Dropout ratio	0.5

Table 11.4: bi-LSTM-CNN base model configuration.

value for the respective hyperparameter showed an increase in F_1-score, then all consequent models were tested with this new hyperparameter value. The starting base model configuration remains the same as notated in table 11.4.

Assuring the generalisation factor of an ML factor is key here since the data used to train this model was purposefully constructed and in turn might differ from production data, a healthy dosage of bias. The first versions of the bi-LSTM-CNN-CRF models were tested with a hyperparameter configuration based on Hovy and Eduard, consisting of 5 epochs, 200 LSTM hidden nodes, learning rate of 0.01. This model was too complex and overfit on the data, determined from the perfect F_1 scores and the low number of epochs. The grid-search results of this model can be found in appendix F, including results of models with stemmed and lemmatised tokens. After trial and error and applying a grid-search (table 11.5), the complexity of the base model was reduced by lowering the number of LSTM hidden nodes to 50 and learning rate of 0.001 with 20 epochs. Two dropout layers were added with a rate of 0.10.

The choice to manually measure the results by analysing the recall and precision scores was deliberate. The total F_1-score of every tested model was always at least above 0.92, while the recall or precision scores of individual named entities were very low. With some understandable exceptions such as testing with lower epochs and higher learning rate. The complete grid-search results of the bi-LSTM-CNN-CRF model can be found in table 11.5. The chosen configuration of the respective hyperparameter is highlighted by a top and bottom dashed border.

► Grid-search and results analysis

The overall grid-search results are very positive, although varied between the named entities. The grid-search results are analysed and interpreted to the following hyperparameter configuration. A higher learning rate shows better results, however, as demonstrated in appendix F, this results in overfitting, mostly due to the low amount of training data. A kernel size of 3 likely has the best results as it captures the two characters surrounding a special character in a given token, without additional noise[4]. TanH shows the best results for the convolutional activation function. The improvement over ReLu, however, is marginal while the computational price compared to ReLu is considerable, therefore ReLu was chosen. The number of convolutional filters was set to 20, a balance of too few feature maps and too many feature maps. Adding the

[4]E. g. the *at*-sign (@) in an e-mail address is always surrounded by two alphanumerical characters, and a the numerical sequences in a phone number are separated by periods, hyphens, or slashes.

dropout layer, with a dropout ratio of 0.10, helped in preventing overfitting, but a too high dropout ratio resetted the neurons too often, resulting in lower performance. Number of LSTM hidden nodes is set to 50, a higher number would result in overfitting due to a too complex model, and a too low number of hidden nodes doesn't capture all the relations between the datapoints.

Additionally, the model is also tested where the tokens were stemmed and lemmatised, the results are listed in tables 11.6 and 11.7 respectively. The models tested with stemmed tokens or lemmatised tokens do not show an significant increase in precision or recall. It is, however, interesting to note that if the data needs to be persisted then stemming will reduce the size of the data significantly while performance.

HP	e-mail			phone			IBAN			pers. ID		
	prec	rec	F_1	prec	recall	F_1	prec	rec	F_1	prec	recall	F_1
Convolution kernel size												
2	0.90	0.97	0.93	0.99	0.99	0.99	0.98	0.67	0.80	0.89	0.97	0.93
3	0.89	0.96	0.93	0.96	0.99	0.97	0.88	0.94	0.91	1.00	0.76	0.86
4	0.89	0.96	0.93	0.99	0.99	0.99	0.85	0.95	0.90	0.98	0.77	0.86
5	0.89	0.95	0.90	0.96	0.96	0.96	0.84	0.96	0.90	1.00	0.64	0.78
Number of convolution strides												
1	0.89	0.98	0.93	0.99	0.99	0.99	0.92	0.96	0.94	1.00	0.51	0.67
2	0.87	0.98	0.92	0.99	1.00	0.99	0.92	0.96	0.94	1.00	0.42	0.59
3	0.85	0.96	0.90	0.99	1.00	0.99	0.88	0.97	0.92	1.00	0.42	0.59
Learning rate[1]												
0.0010	0.90	0.96	0.93	0.99	0.99	0.99	0.87	0.94	0.90	0.98	0.73	0.84
0.0025	0.96	0.98	0.97	0.98	0.98	0.98	0.95	0.98	0.97	0.98	0.94	0.96
0.0050	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.96	0.97	0.99	0.99	0.99
0.0075	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.99	0.99	1.00	0.98	0.99
0.01	0.98	0.99	0.99	0.98	0.98	0.98	0.99	0.98	0.99	1.00	1.00	1.00
Convolution activation function												
Softmax	0.88	0.96	0.91	1.00	1.00	1.00	0.88	0.95	0.91	0.98	0.64	0.77
TanH	0.92	0.97	0.94	0.99	0.99	0.99	0.87	0.94	0.91	1.00	0.88	0.94
ReLu	0.89	0.96	0.93	0.99	1.00	0.99	0.88	0.95	0.91	0.99	0.86	0.92
Number of convolution filters												
10	0.85	0.94	0.89	0.68	0.68	0.68	0.86	0.93	0.89	0.94	0.61	0.74
20	0.88	0.95	0.91	0.99	0.99	0.99	0.87	0.95	0.91	0.99	0.83	0.90
30	0.88	0.97	0.92	0.99	0.99	0.99	0.88	0.95	0.92	1.00	0.60	0.75
Dropout												
0.10	0.90	0.96	0.93	1.00	1.00	1.00	0.90	0.97	0.93	0.98	0.70	0.82
0.25	0.82	0.95	0.88	0.99	0.99	0.99	0.87	0.96	0.91	1.00	0.30	0.46
0.50	0.79	0.92	0.85	1.00	0.99	0.99	0.84	0.95	0.89	1.00	0.42	0.59
LSTM hidden nodes												
25	0.70	0.83	0.76	0.99	0.99	0.99	0.79	0.93	0.85	1.00	0.14	0.25
50	0.86	0.96	0.91	0.99	0.99	0.99	0.85	0.93	0.89	1.00	0.64	0.78
75	0.94	0.98	0.96	0.99	1.00	0.99	0.93	0.96	0.94	1.00	0.95	0.98

[1] The number of epochs remained the same for every learning rate test.

Table 11.5: bi-LSTM-CNN-CRF hyperparameter grid search results

HP	e-mail			phone			IBAN			pers. ID		
	prec	rec	F_1	prec	recall	F_1	prec	rec	F_1	prec	recall	F_1
Convolutional kernel size												
2	0.87	0.96	0.91	0.98	0.99	0.98	0.89	0.94	0.92	0.98	0.64	0.78
3	0.89	0.97	0.93	0.99	1.0	0.99	0.92	0.93	0.92	0.95	0.49	0.80
4	0.89	0.96	0.92	0.98	0.99	0.98	0.87	0.92	0.89	0.92	0.82	0.87
5	0.82	0.96	0.89	0.98	0.99	0.98	0.87	0.89	0.88	0.97	0.39	0.53
Number of convolutional strides												
1	0.92	0.98	0.95	0.98	0.99	0.98	0.93	0.94	0.94	0.98	0.80	0.88
2	0.89	0.97	0.93	0.99	0.99	0.99	0.92	0.96	0.94	0.98	0.60	0.74
3	0.93	0.99	0.96	0.98	0.99	0.98	0.96	0.94	0.95	0.96	0.79	0.86
Number of convolutional filters												
10	0.82	0.90	0.86	0.98	0.94	0.96	0.93	0.96	0.95	1.00	0.60	0.75
20	0.87	0.95	0.91	0.98	0.98	0.98	0.90	0.95	0.92	1.00	0.71	0.83
30	0.91	0.97	0.94	0.99	0.98	0.98	0.92	0.96	0.94	1.00	0.86	0.92

Table 11.6: CNN hyperparameters grid search results of the bi-LSTM-CNN-CRF model with lemmatised tokens.

HP	e-mail			phone			IBAN			pers. ID		
	prec	rec	F_1	prec	recall	F_1	prec	rec	F_1	prec	recall	F_1
Convolutional kernel size												
2	0.90	0.98	0.93	0.98	0.99	0.99	0.83	0.89	0.86	0.96	0.69	0.80
3	0.90	0.95	0.93	0.97	0.99	0.98	0.80	0.85	0.83	0.96	0.85	0.91
4	0.87	0.95	0.91	0.99	0.99	0.99	0.76	0.88	0.82	0.97	0.77	0.86
5	0.87	0.97	0.92	0.99	0.99	0.99	0.85	0.90	0.87	1.00	0.57	0.72
Number of convolutional strides												
1	0.83	0.95	0.89	0.97	0.99	0.98	0.82	0.91	0.86	0.97	0.59	0.73
2	0.86	0.96	0.91	0.97	0.98	0.97	0.74	0.84	0.79	1.00	0.69	0.82
3	0.87	0.96	0.91	0.98	0.99	0.99	0.80	0.88	0.84	0.94	0.62	0.75
Number of convolutional filters												
10	0.77	0.91	0.84	0.89	0.82	0.85	0.52	0.66	0.58	0.97	0.32	0.48
20	0.78	0.93	0.85	0.98	0.98	0.98	0.48	0.79	0.73	1.00	0.24	0.39
30	0.87	0.97	0.91	0.98	0.99	0.99	0.84	0.89	0.87	1.00	0.66	0.80

Table 11.7: CNN hyperparameters grid search results of the bi-LSTM-CNN-CRF model with stemmed tokens.

12

Two-stage feed-forward neural network using engineered features

▸ Synopsis opens by explaining two-stage FNN architecture and reiterates the related works that demonstrate and discuss this approach (§12.1). Each architectural stage is detailed on a per chapter basis (§12.2 and §12.3) and the results of bot models is aggregated, analysed, and discussed (§12.4).

12.1 OVERVIEW

A Feed-forward Neural Network (FNN) is a type of NN where the information only flows forward, unlike RNNs. Consisting of multiple layers, the neurons of each layer connect to all neurons of its adjacent layer(s). FNNs are considered simple architectures as it often does not use any deep learning layers. Relying solely on finding patterns in the given features, instead of understanding and making the features from the data. The biggest benefit is that there are less parameters to train, which reduces training time and storage size. Though often sacrificing on it's accuracy due to a lesser understanding of the data.

Unique identifiers often have a repetitive and distinctive format. Using feature engineering, specific properties of each unique identifier can be taught to the FNN so that it is less reliant on context, which is often limited. Providing more stable features for the algorithm to map input data to an output classification. Two-stage FNN models have been proven successful for various NER tasks [Bot+17]. The used and proven architectures work by first recognising potential candidate entities using a probability score, and second by classifying the candidate entity

[Bot+17] Botha et al., "Natural Language Processing with Small Feed-Forward Networks" to its respective label [BM12; ML16]. Another approach is to recognise candidate entities via clustering[LZ13]. Krishnan and Manning demonstrate that a two-stage model using a neural network and conditional random field can also be used for non-local dependencies[KM06]. The entities this thesis concern itself with however are predominantly locally dependant, this approach is thus not further investigated.

[BM12] Buitinck and Marx, "Two-Stage Named-Entity Recognition Using Averaged Perceptrons"

[ML16] Mozharova and Loukachevitch, "Two-stage approach in Russian named entity recognition"

[LZ13] Liu and Zhou, "Two-stage NER for tweets with clustering"

[KM06] Krishnan and Manning, "An effective two-stage model for exploiting non-local dependencies in named entity recognition"

Training FNN models is relatively fast, as there are no internal preprocessing or feature engineering steps. This is in direct benefit of one sub-requirement, to be quickly re-trainable on new data and deployed to a production environment.

12.2 SCORING ENTITIES AS CANDIDATES

The first model in this two-stage architecture is responsible for selecting possible named entities but not yet classifying them. As a named entity can span multiple tokens, as detailed in chapter §3.2, classifying each possible token, and all its subsequences, in a given document is computationally expensive. To reduce the number of classifications to perform, the first model assigns a score between 0 and 1 to determine how likely a token represents a valid named entity.

The input features are as described in chapter §9.7, where the number of token-at-position features is seven. To recap and summarise: first, a given document is tokenised, and all subsequences of max seven tokens is created as token-at-position features of the given token. See algorithm 5 for implementation details. E. g., all token subsequences of the token *06* in the sentence '*The phone number is 06 1223 4569, the old one is no longer in use.*' are listed in table 12.1. Followed by determining the casings[1] for each token and all token-at-position features. Before the data is fed to the FNN input layer, it first enters a masking layer, which purpose is to ignore all padded features. The output layer uses the cross entropy loss method, assigning a score between 0 and 1 to both binary class outputs.

[1] E. g., is the token capitalised, numerical, uppercase, or lowercase, and whether the token contains the special characters @.

The assigned score for all token subsequences is collected. Only the highest score of any overlapping subsequence is considered for the classification stage, the second FNN model. The result is a list of token sequences from a given document which are deemed likely to be one of the concerned named entities. Reducing the amount of classifications the second-stage of this architecture has to perform.

All results can be found in chapter §12.4, the correct token subsequence is bordered by dashed lines.

			06			
			06	1223		
			06	1223	4569	
			06	1223	4569	,
		is	06			
		is	06	1223		
		is	06	1223	4569	
		is	06	1223	4569	,
	number	is	06			
	number	is	06	1223		
	number	is	06	1223	4569	
	number	is	06	1223	4569	,
phone	number	is	06			
phone	number	is	06	1223		
phone	number	is	06	1223	4569	
phone	number	is	06	1223	4569	,

Table 12.1: Token subsequences example of the token *06* in the sentence '*The phone number is 06 1223 4569, the old one is no longer in use.*'.

Algorithm 5: Creating a list of token subsequences per token

```
  Data: List of tokens $t$
  Result: 2d list of all token subsequences of $t$
1 subsequences = [ ]
2 $size$ = len(tokens) +1
3 if $size < 6$ then $window = size$
4 else window = 6
5 for $i$ in range(size) do
6     for $j$ in $range(i+1, window)$ do
7         subsequences.append(tokens[$i : j$])
```

12.3 TAGGING CANDIDATE ENTITIES

The second model, in this two-stage architecture, is responsible for classifying the supplied candidate entities. Adding a label to the candidate entities found during the recognition stage. The input data is similar to the recognition model of the first-stage and detailed in chapter §9.1.

Each candidate entity is supplied with additional features as described in chapter §9.7. The three preceding token-at-features of the first token in the subsequence are added. And the three proceeding token-at-features of the last token in the subsequence are added. All token features are accompanied by its casings features, as described in chapter

§9.3. Additionally, the candidate entity subsequence length and a bag-of-words of the entity in vector form are created as separate features. Each token in the candidate entity sequence is considered as a separate feature too. The named entities practically never span more than 6 subsequent tokens. Hence, there are six feature inputs available to the FNN model. If there are fewer candidate entity tokens than available input features, then the candidate entity is padded. Thus, every token in the candidate entity subsequence is added as a separate feature.

The various others hyperparameters of this model, such as the amount of layers and neurons, are further discussed in chapter 12.4.

12.4 TWO-STAGE-FNN-BOW NAMED ENTITY RECOGNITION MODEL

The final version of this NER model is a two-stage FNN architecture. First, scoring all tokens and token subsequences for candidate entities. Second, classifying the candidate entities to a named entity or *other* category. The results were expected to be lower than the bi-LSTM-CNN-CRF model, but in turn the performance gains would be improved significantly. However, the results are considerably lower than the aforementioned model. Therefore, the performance gains were not further measured as the model its accuracy is too low. Do note that this model was only tested in a limited form due to time constraint, opposed to the hyperparameter grid-search of the aforementioned model. It is also improbable that a fine-tuned hyperparameter configuration would lead to an acceptable model accuracy. Two different approaches to feature inputs were tested, the primary difference is how the context embeddings are presented as input. The models were tested with the following base hyperparameter configuration: 8 epochs, 8 hidden nodes with ReLu activation, and Softmax as output activation.

13

Custom trained SpaCy NER model

▸ SYNOPSIS This chapter covers the custom training of the SpaCy NER model. First, covering the architecture as is known (§13.1), an exact re-iteration of chapter §7.5. Second, listing the results and the analysis (§13.2). Data processing steps and feature engineering steps are not covered as the model does not provide customisation options.

13.1 SPACY NER ARCHITECTURE - A BLACK BOX

This section is an exact re-iteration of chapter §7.5. As one might defer from the title of this chapter, the architecture of SpaCy's NER model is not published. SPACY is an open source NLP library, written in PYTHON. Providing tokenisers, POS-taggers, embedding strategies, and most importantly a NER model. The exact architecture of the NER model, however, is not published[1]. SPACY NER uses subword features and BLOOM embedding, and a token based deep CNN with residual connections[2] [Das].

[1] As of: 15-06-2020

[2] https://www.youtube.com/watch?v=sqDHBH9IjRU

[Das] Dasagrandhi, *Understanding Named Entity Recognition Pre-Trained Models*

13.2 CUSTOM TRAINED NER RESULTS

The proof-of-concept of this model showed unpromising results. Testing SPACY EN_CORE_WEB_SM pre-trained model also demonstrated its lack, or in some cases its complete inability, of accurately recognising and classifying named entities that contain numerals, such as dates or plain cardinals. However, since the model cannot be tweaked, and therefore did not require new or additional pre-processing steps and feature engineering steps, the model was tested again on the new data (§8). Table 13.1 lists the precision, recall, and F_1-score test results per named entity of this model.

	precision	recall	F_1-score
phone	0.70	0.64	0.67
e-mail	0.77	0.62	0.69
IBAN	0.52	0.72	0.60
pers. ID	0.84	0.42	0.56

Table 13.1: Custom trained SpaCy NER model.

The results are again limited and show a lack of recognising entities containing numerals during manual testing. However, the increased accuracy over the POC model does confirm an increase of data quality and number of samples. Due to time limitations, this model is not further discussed or analysed.

13.2.1 *Separated context features*

This model follows the feature architecture as described in chapter 9. Each context token and entity token is entered as a separate input feature. If there are fewer context tokens or entity tokens, then the respective empty feature inputs are zero padded.

Listed in table 13.2, the results, compared to the bi-LSTM-CNN-CRF model, are lower than expected and the FNN performance increase does not make up for the accuracy difference.

	precision	recall	F_1-score
phone	0.67	0.90	0.77
e-mail	0.76	0.48	0.59
IBAN	0.93	0.80	0.86
pers. ID	0.96	0.56	0.71

Table 13.2: bi-LSTM with CNN layers on token-level results.

13.2.2 *Shared context features*

Using fewer features, the model should understand context better as part of a larger concept than as separate inputs. The separated tokens left and right of the entity tokens are combined and presented as two feature inputs to the model, a left and right context feature, containing multiple embeddings.

The results of the shared context FNN model can be found in table 13.3. The most notable change is that the e-mail named entity recall score improved at the expense of the phone named entity. The model, however, remains underperforming of the bi-LSTM-CNN-CRF model.

	precision	recall	F_1-score
phone	0.67	0.84	0.75
e-mail	0.76	0.56	0.64
IBAN	0.91	0.80	0.85
pers. ID	0.95	0.56	0.70

Table 13.3: bi-LSTM with CNN layers on token-level results.

14

Former implementation and new model comparison

► SYNOPSIS lists the additional steps and methods to evaluate the final bi-LSTM-CNN-CRF model besides the testing subset. First, using public datasets that contain one of the named entities and manually evaluate the results (§14.1.1). Second, comparing the former implementation and the final model by submitting the same production data to both systems and analysing the results (**??**). Last, concluding the results of both tests and discussing a combination of the results (§14.1.4).

14.1 RECOGNITION AND CLASSIFICATION ACCURACY

To determine how the newly developed bi-LSTM-CNN-CRF model compares to the former implementation, the testing subset of the NER model and a production sample set were submitted to both solutions. Both methods are elaborated in the consecutive sections, respectively (§14.1.1 through §14.1.3).

14.1.1 *Validation test*

Because the model was configured based on its performance on the testing data, changing the layers and fine-tuning the hyperparameters, an additional set of data is required to determine that the model was not overfit on the training data. To validate the model, a number of public datasets were separated on a document basis and submitted to the model. The resulting classifications were manually compared to the actual token. This process could unfortunately not be automated, as it would require the knowledge of where each named entity is located. The following datasets were used:

- **Enron**: *This e-mail dataset was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). It contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M e-mails. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation* [KY04].

[KY04] Klimt and Yang, "The Enron Corpus: A New Dataset for Email Classification Research"

- **SMS spam**: *A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC)[Che15], which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available* [AHY11].

[Che15] Chen, "The National University of Singapore SMS Corpus"

[AHY11] Almeida et al., "Contributions to the study of SMS spam filtering"

Because the named entities in these datasets are not annotated, the precision, recall and F_1-score could not be calculated. The result of the bi-LSTM-CNN-CRF model on the public datasets were manually analysed and interpreted by the thesis author. This interpretation was shared and demonstrated to the stakeholders, who could also inject their own documents and experience the results.

14.1.2 *Testing set*

All the samples in the testing subset, that were split from the complete dataset, were submitted to both the former implementation and the NER model as separate documents. Both solutions returned a list of all named entities and the corresponding label found in the given document, along with the exact start character position and the end character position. Each sample in the testing subset has exactly one annotated named entity, marked by the character start position and end character position. Therefore, the result of the former implementation and NER model can be directly compared to the testing subset. As a result, the precision, recall, and F_1 score for both solutions can be calculated and compared.

The former implementation model outperformed the NER model on all named entities with near perfect scores. However, many named entities in the former implementation results had multiple false positive classifications spanning the same token sequence. For some named entities, the former implementation recognised and classified 1.5 times as many token sequences as named entity than exist in the testing set of that named entity[1] The former implementation, also, does not accurately recognise the span of a named entity, often excluding parts of the named entity or including neighbouring unrelated tokens. After adjusting for these errors, where false positives were scored accordingly and overlapping labels were considered incorrect, the overall score changes significantly. The results of the bi-LSTM-CNN-CRF model and the former implementation on testing set can be observed in tables 14.1 and 14.2, respectively.

[1] I. e. the number of false positives for some named entities, by the former implementation, are tremendously high.

	precision	recall	F_1-score
phone	0.68	0.53	0.60
e-mail	0.98	0.99	0.98
IBAN	0.98	0.98	0.99
pers. ID	0.93	0.99	0.96

Table 14.1: Former implementation results on testing set.

	precision	recall	F_1-score
phone	0.93	0.97	0.96
e-mail	0.96	0.91	0.94
IBAN	0.95	0.93	0.94
pers. ID	0.98	0.95	0.96

Table 14.2: bi-LSTM-CNN-CRF results on testing set.

Primarily, the phone number named entity suffers from the error adjustment. There were 587 personal identification named entities in the testing set but 627 were found by the former implementation, these were all partial partial phone numbers. Any token subsequence of a phone number that contained two periods was classified as a personal identifier. This can be observed in a lower precision score for the personal identification named entity and in the recall score of the phone named entity. The NER model performs worse on IBAN named entities and e-mail address named entities, probably due to the lack of diverse data samples for said named entities. All the different IBAN formats that occur in the production set were trained, but the context differs. This shows a clear prioritisation in deducing named entities from context, and not to the same extend the format.

The NER model outperformed the former implementation in two of the named entities. The results of both systems, however, are biased because the NER model was trained on this exact dataset construction and the former implementation was designed around the exact format of the named entities. All samples in the dataset originate from a select few sources, which are similar. Therefore, the production set test is key in determining which system performs better.

14.1.3 *Production set*

A number of production documents, that were ingested in the former implementation, were persisted along with the results. These documents were submitted to the model, and the results were compared to the results of the former implementation. Considering there is no absolute truth to this set, in terms of annotations and labelling, the results could not be calculated entirely automatic. Since all values of the named entities are unique, the results of both systems can be compared by the entity its value. Classifications of the same token sequence span with the same output label were considered correct. This evaluation set was focused on finding the differences between the NER model and the former implementation. Thus correctly classified named entities were not interesting for a comparison and not further considered. Named entity values that exist in only one result map, and named entity values which do occur in both result maps but that have a different label, were manually evaluated.

Similar to the testing set, the NER model showed an improvement in recognising the entire span of a named entity compared to the former implementation, but named entities that were correctly recognised and spanned by the former implementation outperformed the NER model in classifying the named entity accordingly. The former implementation incorrectly classified token sequences to multiple output labels as it considered each token sequence per named entity separately, leading to many false positives and a noise result output. The former implementation recalls more partial named entities, but its lack of recognising the entire sequence and assigning multiple labels damages its usability and adds an overwhelming amount of noise for certain named entities. The NER model did not recognise some named entities at the end of a document, but after placing a token after the named entity, it was classified correctly. This is likely a bias in the NER model, confirming the algorithm that placed the named entity in a given text during the creation of the dataset. This can be prevented by techniques discussed in chapter 15.1. The NER model recalls fewer named entities but is more accurate and provides a comprehensible result, without overlapping labels.

14.1.4 *Analysis and discussion*

In all tests, the bi-LSTM-CNN-CRF model outperformed the former implementation in the recognition and classification of the phone number and personal identification named entities. As the model defers a number of features from the context surrounding the named entities, the model benefits considerably from the embedding techniques and the adaptability factor of machine learning. The IBAN named entity seems to suffer from a lack of diverse contexts, the NER model cannot recognise the texts in which the IBAN named entity can be found in. This problem can be solved by investing in annotating additional data samples by documentalist. If a named entity cannot be classified, as the format differs from the training data, then the new format can be written down as a regular expressions and generated as new format sample. The only catch to generating new format samples is that there must be enough textual samples, as the number of format samples is evenly divided between the number of textual samples. This is an additional benefit over the former implementation, as new undetected formats can be tagged by users and automatically added as new samples to the training data for a future deployment. The lack of textual data, the context surrounding the named entity, can be considered budget problem. At any time, the number of textual samples can be expanded by investing in dedicated documentalists, scouring production data and labelling the token sequences to a corresponding named entity.

14.2 PROCESSING AND EXECUTION TIME

One of the non-functional requirements is that a newly developed solution may not be more computationally expensive than the former implementation, in terms of execution time and necessary hardware.

Both the new NER model and the former implementation were tested on the same NER test data subset and on similar hardware. The total summed execution time was measured during the two stages: loading and pre-processing the data, and applying the recognition and classification system on the data. The total summed execution time for the former implementation is 73 seconds and for the NER model 16 seconds, the NER model is 4.5 times faster than the former implementation. The measurement of the execution time did not take running background processes into account. Adjusting for this, however, would correct the measurement only marginally, not affecting the concluded results. The primary reason that the former implementation is significantly slower is that it applies the different rule based approaches consecutively, having to re-process the entire data multiple times.

Part V

EPILOGUE

15

Conclusion

▶ Synopsis this chapter concludes the achieved results to support the thesis claims, and provides a structured roadmap to further improve the results and state of affairs (§15.1).

The goal of this thesis has been to examine two claims. First, that an NER dataset, of sufficient ML training quality, can be automatically collected and the named entities automatically annotated. Second, that an ML model can recognise and classify unique identifiers as a type of named entities. And we believe that both tasks were achieved amply, with sufficient room for improvement to the data quality and the NER model accuracy, this is further discussed in chapter 15.1. The contribution of this thesis is in the form of a proof demonstration to create an NER dataset, and an implementable advancement in recognising and classifying unique identifiers in documents, elaborated in turn:

- Web scraping to automatically collect data samples, and regular expressions and lookup tables to annotate the named entities.
- CNNs to capture recurring character patterns and the position of special characters in tokens to different feature maps, and engineered casing features that describe unique identifiers.

▶ Automatically collect and annotate NER data
This claim is not just reliant on the findings of its research sub questions, but also on the results of the other claim. Any type of data can be automatically collected, but the concern was the quality of the data and the annotation of the named entity token sequences.

Using web scraping, the textual data was automatically collected and processed. The positive samples primarily origin from online dictionary services, which return sample texts for a given word. The entities themselves were generated using a wide variety of regular expressions or collected from various datasets as samples and added in the sentence construct where the entity naturally occurs. Additionally, the extraordinary large Wikipedia dump dataset was pre-processed and all articles were considered as separate documents. The documents that were proven to not contain a named entity were added as negative samples. Data augmentation was applied to increase the size and diversity of the dataset. Specifically, online editorial sources provided data format consistency and its samples were diverse and of high grammatical quality. The continuous expansion of online web-based editorial data, such as news articles, encyclopedia and dictionaries, makes scraping an extra valuable tool to gradually grow a dataset. The named entities in the documents were annotated using strict regular expressions or look-up tables, as a verifiable absolute truth. This method guarantees that only true positives are added, with only a marginal error ratio of false positives. False negatives can be suppressed, or removed, by weighing each token in a given document using a constraint-driven iterative algorithm, this is considered future work (§15.1.3).

Considering the results of the next claim, the automatically collected and annotated dataset is of sufficient quality to train an NER ML algorithm, sufficient room remains to improve the quality and diversity of the textual samples.

► Recognising and classifying unique identifier named entities
Where traditional rule-based methods had failed, a bi-directional LSTM significantly increased the accuracy in recognising the named entities by providing contextual features. To provide the NER model with a better understanding of the shape of named entities, a CNN layer and engineered token case features were added, increasing the classification accuracy with more engineered features speaking to the format of the named entity. To improve the NER model to recognise the correct and entire span of a named entity, a CRF layer was added to learn the order constraints of the named entities.

The final bi-LSTM-CNN-CRF model achieved an average 0.95 F_1 score on the testing subset and outperforms the former implementation in recognising and classifying the phone number named entity and the personal identifier named entity in all regards. For the remaining IBAN named entity and the e-mail address named entity, the NER model outperforms the former implementation by recognising and classifying the edge-cases but not the base cases. However, the former implementation also suffers from multi-label classifications, where a token sequence can have a wide variety of different annotations. When considering a multi-label classified token as incorrect, the NER model has an increased accuracy score over the former implementation. The trade off is that the NER model recognises fewer named entities in a given document than the former implementation. The stakeholders preferred classification and sequence span accuracy over recognition, but a more preferable solution was concluded. The NER model should supplement the former implementation, rather than replace it. The rule-based methods in the former implementation will be further restricted to reduce the number of false positives and multi-label classification, whereas the NER model will compensate this by recognising and classifying the edge-cases and correcting the annotation span when both systems annotate the same token sequence to the same annotation label.

15.1 FUTURE WORK

Various technologies and techniques will likely improve the results of this thesis or provide further proof to the claims. The following consecutive sub-sections will elaborate on a specific technology and how it will improve the NER model or the data quality.

15.1.1 *Deep contextualized word representations (ELMo)*

The following sentence contains four different meanings for the word *cell*: "Jack while talking over the *cell* phone entered the prison *cell* to extract blood *cell* samples of Jill and made an entry in the excel *cell* about the blood sample collection" - [JSS]. Embedding models such as word2vec and GloVe cannot differentiate between the different meanings of the word *cell*. ELMo creates embeddings also based on the neighbouring words, i. e. the context. The NER model would likely benefit from contextualized word embeddings, to further separate the named entities in the embedding space. Especially the phone number named entity and the e-mail address named entity would benefit from ELMo embeddings, as said entities share many similar contexts. I. e. the following phrases could apply to both named entities: "*My contact details are...*" and "*You can reach me via...*".

[JSS] Jain et al., *How to use ELMo Embedding in Bidirectional LSTM model architecture?*

15.1.2 *Bi-directional Encoder Representations from Transformers (BERT)*

BERT was trained on an extraordinary amount of data of over 100 languages, resulting in a great understanding of natural languages in general and on an individual level for each specific language. The NER model will likely not benefit from higher word embedding accuracy than what ELMo can provide. However, due to the large training data, BERT could probably separate unique identifiers in the embedding space, whereas more traditional and conventional embedding models cannot. This would also make the model more end-to-end, not requiring specific feature engineering.

15.1.3 *Weighing false negatives using a constraint-driven iterative algorithm*

False negatives in an annotated corpus can be suppressed by weighing each token in a given document using a constraint-driven iterative algorithm[May+19]. By weighing the false negatives, the NER algorithm considers those annotated token sequences to a lesser degree. Allowing for the training of a partially annotated dataset, improving the process of automatically annotating a corpus considerably.

[May+19] Mayhew et al., "Named Entity Recognition with Partially Annotated Training Data"

My evaluation

Before this thesis, I had no experience in Natural Language Processing, let alone in Named Entity Recognition. I foresaw a steep learning curve, primarily in learning all the different pre-processing steps and word embedding techniques. Fortunately, the experience was the exact opposite, every time I learned a new technique or technology, a new world of interesting and applicable possibilities opened up to me. And this was exactly my personal pitfall. I constantly wanted to learn everything that I encountered but did not understand. This did allow me to guarantee that the product I delivered is of the highest quality that I can offer, where every single aspect was thoroughly researched with no stones left unturned. But this also works distractious, and I ended up spending costly time and energy on answering sub-research questions that were out of the scope of this research and not defined in the project plan.

My work ethic benefited from the agile approach to the thesis, where the methods and means of achieving the desired results were primarily left to me, of course in consultation and agreement with the stakeholders. Allowing me to redirect or refocus the efforts into a direction that I ought more fruitful. Giving me freedom in conducting in this thesis and allowing me to incorporate new and unconventional architectural decision to achieve the highest possible results, instead of being restricted by maintaining a status quo in sticking to established and proven techniques and wisdom. Consequently, on some occasions I failed to communicate minor, but necessary changes, to some sub-research questions. The changes happened after meeting or consulting with the stakeholders, who were in agreement, but I did not take the step for formal approval.

It was my personal opinion that any form of communication could not have too many details nor a too elaborated context, painting the whole picture to create the clearest possible scenario. But this is an inefficient approach and requires to communicate understand the complete painted picture while following the common thread that is the question. I actively changed this by sharing only the necessary context, the available choices and respective consequences.

The best results are achieved together, by working in and as a team. With many colleagues of my team working from home, or elsewhere in the building, it was critical to maintain contact. Even when there was no new information to share, then exactly that was shared, as it indicates a possible upcoming delay or problem which could affect others. To be a colleague is to be punctual, communicative, and diligent. These traits have always been part of me and define who I am. As I progressed this thesis, I believe to have demonstrated said traits on an exemplary level. The pandemic forced me to put on my A-game, requiring me to improve myself to my best possible capacity.

I was always open to new challenges, but I always knew what I would get myself into and what to expect. But working this thesis required me to learn to become more flexible and adapt to new situations. All kinds of unexpected problems arose during this thesis but no problem is unsolvable. To know when you need help or guidance is when one becomes an effective team player, overcoming any obstacle together.

Bibliography

[AHY11] Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. “Contributions to the study of SMS spam filtering”. In: *Proceedings of the 11th ACM symposium on Document engineering - DocEng ’11* (2011). DOI: 10.1145/2034691.2034742 (cit. on p. 94).

[And16] Martin Andrews. “Compressing Word Embeddings”. In: *Neural Information Processing Lecture Notes in Computer Science* (2016), pp. 413–422. DOI: 10.1007/978-3-319-46681-1_50 (cit. on p. 69).

[Bal+09] Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and James Curran. “Named entity recognition in Wikipedia”. In: (Jan. 2009), pp. 10–18. DOI: 10.3115/1699765.1699767 (cit. on p. 51).

[BSW99] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. “An Algorithm That Learns What‘s in a Name”. In: *Mach. Learn.* 34.1–3 (Feb. 1999), pp. 211–231. ISSN: 0885-6125. DOI: 10.1023/A:1007558221122. URL: https://doi.org/10.1023/A:1007558221122 (cit. on p. 22).

[Bis92] Christopher Bishop. “Pattern recognition and machine learning”. In: (1992). DOI: 10.1016/c2009-0-22409-3 (cit. on p. 59).

[Bot+17] Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan T. McDonald, and Slav Petrov. “Natural Language Processing with Small Feed-Forward Networks”. In: *CoRR* abs/1708.00214 (2017). arXiv: 1708.00214. URL: http://arxiv.org/abs/1708.00214 (cit. on pp. 32, 48, 85).

[BM12] Lars Buitinck and Maarten Marx. “Two-Stage Named-Entity Recognition Using Averaged Perceptrons”. In: *Natural Language Processing and Information Systems Lecture Notes in Computer Science* (2012), pp. 171–176. DOI: 10.1007/978-3-642-31178-9_17 (cit. on p. 86).

[Cre] *CRF Layer on the Top of BiLSTM - 1*. 2017. URL: https://createmomo.github.io/2017/09/12/CRF_Layer_on_the_Top_of_BiLSTM_1/ (cit. on p. 79).

[Che15] Kan Min-Yen Chen T. “The National University of Singapore SMS Corpus”. In: 2015. DOI: https://doi.org/10.25540/WVM0-4RNX (cit. on p. 94).

[CJ20] Davide Chicco and Giuseppe Jurman. “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”. In: *BMC Genomics* 21.1 (2020). DOI: 10.1186/s12864-019-6413-7 (cit. on p. 40).

[CR98] N. Chinchor and P. Robinson. “Appendix E: MUC-7 Named Entity Task Definition (version 3.5)”. In: *Seventh Message Understanding Conference (MUC-7): Proceedings of Conference Held 1028, in Fairfax, Virginia April 29-May 1, 1998*. 1998. URL: https://www.aclweb.org/anthology/M98- (cit. on pp. 5, 6, 19, 51).

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and accurate deep network learning by exponential linear units (ELUs)”. In: *arXiv preprint arXiv:1511.07289* (Nov. 2015). URL: https://arxiv.org/abs/1511.07289 (cit. on p. 121).

[CS02] Michael Collins and Yoram Singer. “Unsupervised Models for Named Entity Classification”. In: *Proceedings of the 1999 Joint SIGDAT Conference on EMNLP and VLC* (Dec. 2002) (cit. on p. 22).

[Das] Sai Dasagrandhi. *Understanding Named Entity Recognition Pre-Trained Models*. URL: https://blog.vsoftconsulting.com/blog/understanding-named-entity-recognition-pre-trained-models (cit. on pp. 49, 89).

[Kdn] *Deep Learning for Natural Language Processing (NLP) – using RNNs & CNNs*. URL: https://www.kdnuggets.com/2019/02/deep-learning-nlp-rnn-cnn.html (cit. on p. 31).

[Den+10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “ImageNet: Constructing a large-scale image database”. In: *Journal of Vision* 9.8 (2010), pp. 1037–1037. ISSN: 1071-5797. DOI: 10.1167/9.8.1037 (cit. on p. 3).

[Der16] Leon Derczynski. “Complementarity, F-score, and NLP Evaluation”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 261–266. URL: https://www.aclweb.org/anthology/L16-1040 (cit. on p. 40).

[Der+17] Leon Derczynski, Eric Nichols, Marieke Van Erp, and Nut Limsopatham. “Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition”. In: *Proceedings of the 3rd Workshop on Noisy User-generated Text* (2017). DOI: 10.18653/v1/w17-4418 (cit. on p. 51).

[DJ99] Włodzisław Duch and Norbert Jankowski. “Survey of neural transfer functions”. In: *Neural Computing Surveys* 2.1 (1999), pp. 163–212. URL: ftp://ftp.icsi.berkeley.edu/pub/ai/jagota/vol2_6.pdf (cit. on p. 121).

[Dug+01] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. “Incorporating Second-Order Functional Knowledge for Better Option Pricing”. In: *Advances in Neural Information Processing Systems 13 (NIPS)*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 472–478. URL: http://papers.nips.cc/paper/1920-incorporating-second-order-functional-knowledge-for-better-option-pricing.pdf (cit. on p. 121).

[GR10] J. Gants and D. Reinsel. *The digital Universe Decade, Are you Ready?* 2010 (cit. on p. 3).

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks.” In: *Aistats*. Vol. 15. 106. 2011, p. 275. URL: http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf (cit. on p. 121).

[GE08] Yoav Goldberg and Michael Elhadad. “splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications”. In: *Proceedings of ACL-08: HLT, Short Papers*. Columbus, Ohio: Assocation for Computational Linguistics, June 2008, pp. 237–240. URL: https://www.aclweb.org/anthology/p08-2060 (cit. on p. 3).

[Goo+13] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. “Maxout networks.” In: *ICML* 28.3 (2013), pp. 1319–1327. URL: http://www.jmlr.org/proceedings/papers/v28/goodfellow13.pdf (cit. on p. 121).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 23, 57).

[GDIV09] Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian. “Streaming for large scale NLP: Language Modeling”. In: *Proceedings of Human Language Technologies; The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado: Association for computational Linguistics, June 2009, pp. 512–520. URL: https://www.aclweb.org/anthology/N09-1058 (cit. on p. 3).

[Gre+17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232 (cit. on p. 26).

[Hac+13] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. “Evaluating Entity Linking with Wikipedia”. In: *Artificial Intelligence* 194 (2013), pp. 130–150. DOI: 10.1016/j.artint.2012.04.005 (cit. on p. 30).

[HSZ11] Xianpei Han, Le Sun, and Jun Zhao. “Collective Entity Linking in Web Text: A Graph-Based Method”. In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’11. Beijing, China: Association for Computing Machinery, 2011, pp. 765–774. ISBN: 9781450307574. DOI: 10.1145/2009916.2010019. URL: https://doi.org/10.1145/2009916.2010019 (cit. on p. 30).

[HC17] David Hand and Peter Christen. "A note on using the F-measure for evaluating record linkage algorithms". In: *Statistics and Computing* 28.3 (2017), pp. 539–547. DOI: 10.1007/s11222-017-9746-6 (cit. on p. 40).

[Hay] Safdar Hayat. *Phone Numbers Classification (PNC) with Feed-forward Neural Networks*. URL: https://www.academia.edu/8771009/Phone_Numbers_Classification_PNC_with_Feed-forward_Neural_Networks (cit. on p. 32).

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *International Conference on Computer Vision (ICCV)*. Feb. 2015, pp. 1026–1034. URL: https://arxiv.org/abs/1502.01852 (cit. on p. 121).

[Her+06] Soto Herranz, Raquel Martínez-Unanue, Arantza Casillas, and Víctor Fernández. "Multilingual News Document Clustering: Two Algorithms Based on Cognate Named Entities". In: vol. 4188. Sept. 2006, pp. 165–172. DOI: 10.1007/11846406_21 (cit. on p. 29).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 26).

[HJ15] Matthew Honnibal and Mark Johnson. "An Improved Non-monotonic Transition System for Dependency Parsing". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1373–1378. URL: https://aclweb.org/anthology/D/D15/D15-1162 (cit. on p. 48).

[HE16] Hovy and Eduard. *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. 2016. URL: https://arxiv.org/abs/1603.01354 (cit. on pp. 31, 32, 75, 80).

[HR18] Jeremy Howard and Sebastian Ruder. "Universal Language model Fine-tuning for Text Classification". In: *CoRR* abs/1801.06146 (2018). arXiv: 1801.06146. URL: https/arxiv.org/abs/1801.06146 (cit. on p. 3).

[JSS] Ankit Jain, Divya Sriadibhatla, and Kumar Sai Surathu. *How to use ELMo Embedding in Bidirectional LSTM model architecture?* URL: https://insights.insofe.com/index.php/2020/02/11/how-to-use-elmo-embedding-in-bidirectional-lstm-model-architecture/ (cit. on p. 103).

[JDS11] H. Jégou, M. Douze, and C. Schmid. "Product Quantization for Nearest Neighbor Search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1 (2011), pp. 117–128 (cit. on p. 69).

[Jou+16] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. "FastText.zip: Compressing text classification models". In: *CoRR* abs/1612.03651 (2016). arXiv: 1612.03651. URL: http://arxiv.org/abs/1612.03651 (cit. on p. 70).

[KG16] Eliyahu Kiperwasser and Yoav Goldberg. "Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations". In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 313–327. DOI: 10.1162/tacl_a_00101. URL: https://www.aclweb.org/anthology/Q16-1023 (cit. on p. 75).

[KY04] Bryan Klimt and Yiming Yang. "The Enron Corpus: A New Dataset for Email Classification Research". In: *Proceedings of the 15th European Conference on Machine Learning*. ECML'04. Pisa, Italy: Springer-Verlag, 2004, pp. 217–226. ISBN: 3540231056. DOI: 10.1007/978-3-540-30115-8_22. URL: https://doi.org/10.1007/978-3-540-30115-8_22 (cit. on p. 94).

[KM06] Vijay Krishnan and Christopher D. Manning. "An effective two-stage model for exploiting non-local dependencies in named entity recognition". In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL 06* (2006). DOI: 10.3115/1220175.1220316 (cit. on p. 86).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25 (NIPS)*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (cit. on p. 121).

[KS02] V. Kurkova and M. Sanguineti. “Comparison of worst case errors in linear and neural network approximation”. In: *IEEE Transactions on Information Theory* 48.1 (Jan. 2002), pp. 264–275. ISSN: 0018-9448. DOI: 10.1109/18.971754. URL: http://ieeexplore.ieee.org/abstract/document/971754/ (cit. on p. 121).

[LMS05] P. Leach, M. Mealling, and R. Salz. “A Universally Unique IDentifier (UUID) URN Namespace”. In: (2005). DOI: 10.17487/rfc4122 (cit. on p. 20).

[LW09] Dekang Lin and Xiaoyun Wu. “Phrase clustering for discriminative learning”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - ACL-IJCNLP 09* (2009). DOI: 10.3115/1690219.1690290 (cit. on p. 4).

[LZ13] Xiaohua Liu and Ming Zhou. “Two-stage NER for tweets with clustering”. In: *Information Processing & Management* 49.1 (2013), pp. 264 –273. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2012.05.006. URL: http://www.sciencedirect.com/science/article/pii/S0306457312000714 (cit. on p. 86).

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. ICML*. Vol. 30. 1. 2013. URL: https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf (cit. on p. 121).

[Man+14] Christpoher D. Manning, Muhai Surdeaunu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. “The Stanford CoreNLP Natural Language Processing Toolkit”. In: 2014, pp. 55–60. URL: https://www.aclweb.org/anthology/P/P14/P14-5010 (cit. on p. 30).

[May+19] Stephen Mayhew, Snigdha Chaturvedi, Chen-Tse Tsai, and Dan Roth. “Named Entity Recognition with Partially Annotated Training Data”. In: Jan. 2019, pp. 645–655. DOI: 10.18653/v1/K19-1060 (cit. on p. 104).

[MP43] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133 (cit. on p. 121).

[McN+11] Paul McNamee, James Mayfield, Dawn Lawrie, Douglas Oard, and David Doermann. “Cross-Language Entity Linking”. In: *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, Nov. 2011, pp. 255–263. URL: https://www.aclweb.org/anthology/I11-1029 (cit. on p. 30).

[Mik+18] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. “Advances in Pre-Training Distributed Word Representations”. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018 (cit. on p. 36).

[Mik+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546 (cit. on p. 36).

[Mil+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. “Introduction to WordNet: An On-line Lexical Database”. In: *International Journal of Lexicography* 3.4 (1990), pp. 235–244. DOI: 10.1093/ijl/3.4.235 (cit. on p. 63).

[ML16] Valerie Mozharova and Natalia Loukachevitch. “Two-stage approach in Russian named entity recognition”. In: *2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT)* (2016). DOI: 10.1109/fruct.2016.7584769 (cit. on p. 86).

[Not+13] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. “Learning multilingual named entity recognition from Wikipedia”. In: *Artificial Intelligence* 194 (2013), pp. 151–175. DOI: 10.1016/j.artint.2012.03.006 (cit. on p. 4).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162 (cit. on p. 31).

[Pet+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. “Deep contextualized word representations”. In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: https://arxiv.org/abs/1802.05365 (cit. on p. 3).

[Por] Martin Porter. *A language for stemming algorithms*. URL: http://snowball.tartarus.org/texts/introduction.html (cit. on p. 62).

[Chr] *Product Quantizers for k-NN*. 2017. URL: https://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1/ (cit. on pp. 70, 71).

[R.+16] Pranav R., Jian Z., Konsten L., and Percy L. “SQuAD: 100,000+ Questions for machine comprehension of text”. In: *Proceedings of the 2015 Conference of Empirical Methods in Natural Language Processing* (2016). DOI: 10.18653/v1/d16-1264 (cit. on p. 30).

[Rab19] Shahariar Rabby. *What is NLP & What Do NLP Scientists Do?* 2019. URL: https://towards.datascience.com/whatnlpscientistsdo-905aa987c5c0 (cit. on p. 62).

[Ram18] Aiswarya Ramachandran. *NLP Guide: Identifying Part of Speech Tags using Conditional Random Fields*. 2018. URL: https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31 (cit. on p. 63).

[RM95] Lance Ramshaw and Mitch Marcus. “Text Chunking using Transformation-Based Learning”. In: *Third Workshop on Very Large Corpora*. 1995. URL: https://www.aclweb.org/anthology/W95-0107 (cit. on p. 22).

[RRM06] Roni Romano, Lior Rokach, and Oded Mainon. “Automatic Discovery of Regular Expression Patterns Representing Negated Findings in Medical Narrative Reports”. In: *Next Generation Information Technologies and Systems*. Ed. by Opher Etzion, Tsvi Kuflik, and Amihai Motro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 300–3011. ISBN: 978-3-540-25473-4 (cit. on p. 30).

[Rue14] Tom Ruettet. *Corpora versus datasets*. 2014. URL: https://corpuslinguisticmethods.wordpress.com/2013/12/28/corpora-versus-datasets/ (cit. on pp. 21, 52).

[SCAG13] Daniel Sanchez-Cisneros and Fernando Aparicio Gali. In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, June 2013, pp. 622–627. URL: https://aclweb.org/anthology/S13-2104 (cit. on p. 4).

[SM03] Erik F. Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 shared task”. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 -* (2003). DOI: 10.3115/1119176.1119195 (cit. on pp. 46, 51).

[Sta] NLP Stanford. *stemming and lemmatization*. URL: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (cit. on p. 62).

[Ste+13] Ralf Steinberger, Maud Ehrmann, Júlia Pajzs, Mohamed Ebrahim, Josef Steinberger, and Marco Turchi. “Multilingual Media Monitoring and Text Analysis - Challenges for Highly Inflected Languages”. In: *TSD*. 2013 (cit. on p. 29).

[Tbe18] Tbennun. *tbennun/keras-bucketed-sequence*. 2018. URL: https://github.com/tbennun/keras-bucketed-sequence (cit. on p. 72).

[Goo] *The Machine Learning Behind Android Smart Linkify*. 2018. URL: https://ai.googleblog.com/2018/08/the-machine-learning-behind-android.html (cit. on p. 32).

[TKSDM03] Erik F. Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL ’03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 142–147. DOI: 10.3115/1119176.1119195. URL: https://doi.org/10.3115/1119176.1119195 (cit. on pp. 22, 31).

[Wei+19] Wei, Jason, Zou, and Kai. *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*. 2019. URL: https://arxiv.org/abs/1901.11196 (cit. on p. 58).

[Wora] *What is WordNet?* URL: https://wordnet.princeton.edu/ (cit. on p. 63).

[Worb] *WordHippo!* *https://www.wordhippo.com/* (cit. on p. 52).

[You] *Your Dictionary* *https://www.yourdictionary.com/* (cit. on p. 52).

[Zaf+19] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. “Q8BERT: Quantized 8Bit BERT”. In: *ArXiv* abs/1910.06188 (2019) (cit. on p. 70).

[Zha+16] Zhang, Xiang, Zhao, Junbo, and Yann. *Character-level Convolutional Networks for Text Classification*. 2016. URL: https://arxiv.org/abs/1509.01626 (cit. on p. 58).

[ZW15] Ye Zhang and Byron C. Wallace. “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification”. In: *CoRR* abs/1510.03820 (2015). arXiv: 1510.03820. URL: http://arxiv.org/abs/1510.03820 (cit. on p. 31).

[Sci] *sklearn.metrics.f1_score*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (cit. on p. 41).

Part VI

APPENDIX

A

Non-functional requirements

A.1 ERROR HANDLING AND LOGGING

The software to load and pre-process the data, and train the NER model, was meticulously written with error handling and logging in mind. Depending on the number of data samples, the pre-processing steps and especially the ML model training can take a long total execution time. The focus was on preventing fatal crashes so that the respective process can continue uninterrupted. E. g. if a datapoint in a given sample contains an incorrect data type, then the entire data sample is escaped and not added to the dataset. The software was written in the scripting language PYTHON, and functions were only used for repetitive tasks which LIST COMPREHENSIONS could not achieve, therefore, the error handling mostly consisted of TRY-EXCEPT-FINALLY blocks. All intermediate steps from collecting raw data, to annotating entities, to model training were thoroughly logged. The user can select different log levels (verbose, info, error) which determine the frequency and detail of individual or sectional logs.

A.2 EXPORTING MODEL AND PORTABILITY

The NER model was written in PYTHON and can be exported to any language that can recreate the architecture of the model and import its learned parameters from an .HDF5 file. This also enables transfer learning, specifically to expand the number of output classifications without having to re-train all parameters. The portability of the model also makes it operating system independent. The pre-processing is rather simple, with only basic data manipulation, which can thus be re-written to any kind of system.

A.3 PROCESS SECURITY

The model can be containerized, such as in Docker, which guarantees that the model processes are isolated. In agreement with the stakeholders, it was also determined that the framework used to develop the model, in combination with a containerized environment, provides enough security and process isolation.

A.4 SOFTWARE DOCUMENTATION

The pre-processing steps and model architecture was written in Python Jupyter notebooks. Every code cell, which can be executed independently, was meticulously documented. Every executable Python cell was prepended with a markdown cell which detailed the required input, the overarching purpose of the executable cell, and the expected output. This created a (chrono)logical order of executable processes. Every line of code, except for assignments, declarations, and basic operations, was documented with comments. Particularly, multi-level list comprehensions were explained.

B

Used regular expressions to capture named entities

B.1 PHONE NUMBERS

Example reverse output	regular expression
799 − 059 − 2817	r”\^[2-9]\2-\3-\4\$”
(082)427 − 7838	r”((\(\3\) ?)|(\3-))?\3-\4”
4 − (2744842925	r”^([0-9](|-)?)?(?[0 − 9]3?|[0-9]3)(|-)?([0-9]3(|-)?[0-9]4)$”
1300314940	r”(^1300{6}$)\|(6̣$)|(^0[2|3|7|8]{1}[0-9]{8}$)\|(^13{4}$)|(^042,3̣{6}$)”1
71653 − 3 − 54234 − 23 − 35	r”(\(?\+?[0-9]*\)?)?[0-9_\- \(\)]*$”
(+27)(0)222438750	r”(^\+[0-9]{2}|^\+[0-9]{2}\(0\)|\(0\)|^00[0-9]{2}$\|[0-9\-\s]{10}$)”

Table B.1: List of regular expressions used to capture phone numbers.

B.2 EMAIL ADDRESSES

Example reverse output	regular expression
rlueQ257Dy@tNmawGPK.eod	r”^\w+@[a-zA-Z_]+?[a-zA-Z]{2,3}$”
WG3WTY7gz-qoRTttTyT1nZ@3z.P	r”^\w+[\w-]*\w+((-\w+)|(\w*))[a-z]{2,3}$”
My”9 7c<va%wb{9o&gt;0̃c\]:no$Cq	r”&lt;[^&gt;]*[\s]*=[\s]*&quot;?[^\w_]*&;?[^&gt;]*&gt;”

Table B.2: List of regular expressions used to capture e-mail addresses.

C

Used context-free grammars to generate sentences

C.1 POSTIVIE SAMPLE CFG'S

```
INTERTWINE → 'because '| 'thats why' | 'otherwise '
COMMUNICATION → 'communicated ' | 'said ' | 'said ' | 'told ' | 'know
DEVICE → 'device ' | 'phone ' | 'gsm' | 'mobile ' | 'mobile phone ' | 'smartphone '
NUMBER → 'number ' | 'digits ' | 'combination ' | 'calling number '
PERSON → 'he ' | 'she ' | 'they ' | 'him ' | 'her '
ACTION → 'gave ' | 'handed over ' | 'sent ' | 'tried ' | 'called ' | 'used ' | 'changed '
TENSES → 'were ' | 'was ' | 'is ' | 'used to be ' | 'is currently ' | 'was currently '
STATE → 'not ' | 'temporarily ' | 'momentarily ' | 'for a while '
ONLINE → 'available ' | 'reachable ' | 'picking up ' | 'online ' | 'in reach '

S → PERSON STATE ACTION PERSON DEVICE [entity] INTERTWINE PERSON STATE ONLINE
```

```
S → NP VP
NP → Nom | PropN | N
Nom → Adj Nom | N
VP → V Adj | V P [entity] NP | V P [entity] S | V P [entity] NP PP
VP → V Adj | V N
PP → P
PropN → 'him ' | 'her ' | 'them '
Det → 'the ' | 'a '
N → 'person ' | 'he ' | 'she ' | 'them '
Adj → 'quick ' | 'long ' | 'fast ' | 'short '
V → 'calls ' | 'called ' | 'reaches ' | 'phones ' | 'contacted ' | 'messaged '
P → 'on '
```

```
INTERTWINE → 'because '| 'thats why' | 'otherwise '
COMMUNICATION → 'communicated ' | 'said ' | 'said ' | 'told ' | 'know
DEVICE → 'device ' | 'phone ' | 'gsm' | 'mobile ' | 'mobile phone ' | 'smartphone '
NUMBER → 'number ' | 'digits ' | 'combination ' | 'calling number '
PERSON → 'he ' | 'she ' | 'they ' | 'him ' | 'her '
ACTION → 'gave ' | 'handed over ' | 'sent ' | 'tried ' | 'called ' | 'used ' | 'changed '
TENSES → 'were ' | 'was ' | 'is ' | 'used to be ' | 'is currently ' | 'was currently '
STATE → 'not ' | 'temporarily ' | 'momentarily ' | 'for a while '
ONLINE → 'available ' | 'reachable ' | 'picking up ' | 'online ' | 'in reach '

S → DEVICE 'and ' NUMBER [entity] PERSON ACTION TENSES STATE ONLINE
```

FIGURE C.1: CFG rules to generate a sentence mentioning or referring to a phone number as a positive data sample.

C.2 NEGATIVE SAMPLE CFG'S

```
S → NP VP
NP → Nom | PropN | N
Nom → Adj Nom | N
VP → V Adj | V P [entity] NP | V P [entity] S | V P [entity] NP PP
VP → V Adj | V N
PP → P
PropN → 'him' | 'her' | 'them'
Det → 'the' | 'a'
N → 'person' | 'he' | 'she' | 'them'
Adj → 'quick' | 'long' | 'fast' | 'short'
V → 'confirmation' | 'authentication' | 'card' | 'two factor'
P → 'code' | 'number'
```

```
METHOD → 'created'| 'given' | 'personal'
TYPE → 'confirmation'| 'authentication' | 'card' | 'two factor'
NUMBER → 'number' | 'code' | 'ID'
VALIDITY → 'valid' | 'usable' | 'accessible'
WHEN → 'until' | 'now' | 'till' | 'from'
TIME → 'today' | 'tomorrow' | 'from now on' | 'week' | 'currently'

S → METHOD B ]entity] VALIDITY WHEN TIME
B → 'company' | TYPE | TYPE NUMBER
```

```
METHOD → 'created'| 'given' | 'personal'
TYPE → 'confirmation'| 'authentication' | 'card' | 'two factor'
NUMBER → 'number' | 'code' | 'ID'
VALIDITY → 'valid' | 'usable' | 'accessible'
WHEN → 'until' | 'now' | 'till' | 'from'
TIME → 'today' | 'tomorrow' | 'from now on' | 'week' | 'currently'

S → TIME WHEN VALIDITY B [entity] METHOD WHEN TIME
S → METHOD B [entity] VALIDITY WHEN TIME
B → TYPE | TYPE NUMBER
```

FIGURE C.2: CFG rules to generate a sentence mentioning or referring to entities similar to phone numbers as a negative data sample.

D

Equations of activation functions

Name	Function $\varphi(x)$	Range of Values	Used by
Sign function†	$\begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$	$\{-1, 1\}$	Kurkova and Sanguineti
Heaviside step function†	$\begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$	$\{0, 1\}$	McCulloch and Pitts
Logistic function	$\frac{1}{1+e^{-x}}$	$[0, 1]$	Duch and Jankowski
ReLU†	$\max(0, x)$	$[0, +\infty)$	Krizhevsky et al.
LReLU†[1] (PReLU	$\varphi(x) = \max(\alpha x, x)$	$(-\infty, +\infty)$	Maas et al.; He et al.
Softplus	$\log(e^x + 1)$	$(0, +\infty)$	Dugas et al.; Glorot et al.
ELU	$\begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$	$(-\infty, +\infty)$	Clevert et al.
Softmax‡	$o(\mathrm{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$	$[0, 1]^K$	Krizhevsky et al.
Maxout‡	$o(\mathrm{x}) = \max_{x \in \mathrm{x}} x$	$(-\infty, +\infty)$	Goodfellow et al.

TABLE D.1: List of activation functions, its respective equation and range of values and the papers it used in.

E

Wikipedia dump dataset pre-processing script

To train the embedding layer, detailed in chapter §5.5, the Wikipedia dump dataset was pre-processed as denoted in algorithm 6, developed by Matt Mahoney.

source: `http://mattmahoney.net/`.

Algorithm 6: Pearl script to pre-process the Wikipedia dump dataset.

```
Data: Wikipedia dump dataset
Result: Wikipedia articles without xml or html tags
begin
    $/=”>”;
    while <> do
        if /<text / then
            $text=1;
        if (/#redirect/i) then
            $text=0;
        if $text then
            if /<text>/ then
                $text=0;
            s/<.*>//;
            s/&amp;/&/g;
            s/&lt;/</g;
            s/&gt;/>/g;
            s/<ref[^<]*<\/ref>//g;
            s/<[^>]*>//g;
            s/\[http:[^] ]*/[/g;
            s/\|thumb//ig;
            s/\|left//ig;
            s/\|right//ig;
            s/\|\d+px//ig;
            s/\[\[image:[^\[\]]*\|//ig;
            s/\[\[category:([^|\]]*)[^]]*\]\]/[[$1]]/ig;
            s/\[\[[a-z\-]*:[^\]]*\]\]//g;
            s/\[\[[^\|\]]*\|/[[/g;
            s/[^]*//g;
            s/[^]*//g;
            s/\[//g;
            s/\]//g;
            s/&[^;]*;/ /g;
            $_=” $_ ”;
            s/”+//g;
            tr/A-Za-z0-9˝”!?/ /cs;
            s/ ([,.!?]) /$1 /g;
            chop;
            print $_;
```

F

Additional bi-LSTM-CNN grid-search testing

The first versions of the bi-LSTM-CNN-CRF models were tested with a base hyperparameter configuration of 5 epochs, 200 LSTM hidden nodes, learning rate of 0.01, and 0.1 dropout. As can be observed from the results, this model was too complex and overfit on the data. The complete results can be found in table F.1. The results of the models with stemmed and lemmatised tokens are listed in tables F.2 and F.3 respectively.

HP	e-mail			phone			IBAN			pers. ID		
	prec	rec	F_1	prec	recall	F_1	prec	rec	F_1	prec	recall	F_1
Number of epochs												
2	0.00	0.00	0.00	0.63	0.56	0.59	0.66	0.84	0.74	0.00	0.00	0.00
3	0.99	0.95	0.97	0.77	0.88	0.82	0.75	0.91	0.82	0.83	0.35	0.49
4	0.98	0.96	0.97	0.83	0.93	0.88	0.89	0.91	0.90	0.96	0.55	0.70
5	0.97	0.99	0.98	0.79	0.94	0.86	0.85	0.83	0.84	1.00	0.28	0.44
6	0.97	1.00	0.99	0.86	0.97	0.91	0.91	0.91	0.91	0.99	0.78	0.87
7	0.98	1.00	0.99	0.91	0.98	0.94	0.92	0.87	0.89	1.00	0.88	0.94
8	0.97	1.00	0.98	0.95	0.98	0.96	0.97	0.94	0.96	1.00	0.93	0.96
9	0.98	1.00	0.99	0.95	0.98	0.97	0.98	0.94	0.96	0.98	0.95	0.96
Convolution kernel size												
2	0.97	0.94	0.95	0.87	0.94	0.90	0.90	0.96	0.93	0.95	0.77	0.85
3	1.00	1.00	1.00	0.92	0.97	0.94	0.94	0.95	0.95	0.96	0.83	0.89
4	0.98	0.96	0.97	0.89	0.95	0.92	0.87	0.95	0.91	0.96	0.83	0.89
5	0.98	0.96	0.97	0.89	0.95	0.92	0.87	0.95	0.91	0.96	0.83	0.89
Number of convolution strides												
1	0.97	0.98	0.98	0.87	0.97	0.92	0.91	0.91	0.91	0.96	0.69	0.80
2	1.00	1.00	1.00	0.92	0.96	0.94	0.92	0.95	0.93	1.00	0.84	0.92
3	1.00	1.00	1.00	0.93	0.99	0.96	0.95	0.95	0.95	0.99	0.78	0.87
Learning rate												
0.001[1]	0.47	0.46	0.46	0.34	0.47	0.40	0.38	0.72	0.50	0.00	0.00	0.00
0.01	0.95	0.99	0.97	0.89	0.97	0.93	0.90	0.89	0.89	1.00	0.74	0.85
Convolution activation function												
softmax	0.98	1.00	0.99	0.92	0.95	0.93	0.88	0.95	0.91	1.00	0.94	0.97
tanh	0.98	0.98	0.98	0.92	0.96	0.94	0.86	0.90	0.88	1.00	0.94	0.97
ReLu	1.00	1.00	1.00	0.91	0.97	0.94	0.94	0.92	0.93	0.99	0.72	0.83
Number of convolution filters												
10	0.98	0.85	0.91	0.91	0.93	0.92	0.88	0.95	0.91	0.99	0.95	0.97
20	0.95	0.98	0.97	0.94	0.97	0.95	0.88	0.92	0.90	0.96	0.95	0.95
30	0.95	0.98	0.97	0.88	0.96	0.92	0.93	0.94	0.94	0.96	0.68	0.79
LSTM hidden nodes												
100	1.00	1.00	1.00	0.94	0.97	0.95	0.95	0.93	0.94	0.98	0.93	0.95
150	0.98	0.99	0.98	0.91	0.95	0.93	0.91	0.90	0.90	0.93	0.88	0.90
200	0.96	0.98	0.97	0.89	0.98	0.93	0.93	0.94	0.94	0.99	0.83	0.90

[1] The number of epochs remained the same for every learning rate test.

Table F.1: bi-LSTM-CNN-CRF hyperparameter grid search results

HP	e-mail			phone			IBAN			pers. ID		
	prec	rec	F_1	prec	recall	F_1	prec	rec	F_1	prec	recall	F_1
Convolutional kernel size												
2	0.99	1.00	1.00	0.87	0.99	0.93	0.92	0.90	0.91	1.00	0.62	0.76
3	0.97	0.99	0.98	0.95	0.96	0.95	0.95	0.95	0.95	0.98	0.94	0.96
4	0.33	1.00	1.00	0.89	0.96	0.92	0.93	0.95	0.94	1.00	0.78	0.88
5	0.99	1.00	1.00	0.93	0.97	0.95	0.95	0.93	0.94	1.00	0.87	0.93
Number of convolutional strides												
1	0.98	0.99	0.98	0.85	0.97	0.90	0.93	0.93	0.93	1.00	0.55	0.71
2	0.99	1.00	1.00	0.82	0.97	0.89	0.91	0.84	0.87	1.00	0.46	0.63
3	0.99	1.00	1.00	0.87	0.99	0.93	0.97	0.94	0.96	1.00	0.52	0.68
Number of convolutional filters												
10	0.75	0.35	0.47	0.90	0.99	0.94	0.96	0.93	0.94	0.97	0.68	0.80
20	0.95	0.99	0.97	0.88	0.97	0.92	0.91	0.89	0.90	0.99	0.71	0.82
30	0.99	0.99	0.99	0.90	0.98	0.94	0.94	0.91	0.92	0.95	0.75	0.84

Table F.2: CNN hyperparameters grid search results of the bi-LSTM-CNN-CRF model with lemmatised tokens

HP	e-mail			phone			IBAN			pers. ID		
	prec	rec	F_1	prec	recall	F_1	prec	rec	F_1	prec	recall	F_1
Convolutional kernel size												
2	1.00	1.00	1.00	0.82	0.97	0.89	0.81	0.78	0.79	1.00	0.58	0.73
3	0.99	1.00	0.99	0.78	0.97	0.86	0.81	0.77	0.79	1.00	0.38	0.55
4	0.98	0.99	0.99	0.92	0.95	0.94	0.86	0.86	0.86	0.92	0.97	0.94
5	0.98	0.98	0.98	0.58	0.97	0.91	0.81	0.80	0.80	0.99	0.90	0.94
Number of convolutional strides												
1	0.99	0.99	0.99	0.90	0.97	0.93	0.87	0.89	0.88	0.95	0.92	0.94
2	0.98	1.00	0.99	0.88	0.95	0.91	0.77	0.78	0.77	0.93	0.89	0.91
3	0.98	0.99	0.99	0.92	0.96	0.94	0.87	0.88	0.84	0.89	0.87	0.88
Number of convolutional filters												
10	0.98	0.91	0.94	0.89	0.96	0.93	0.82	0.82	0.83	0.98	0.93	0.96
20	0.96	0.86	0.91	0.89	0.97	0.93	0.76	0.77	0.76	0.97	0.85	0.90
30	1.00	0.99	1.00	0.87	0.97	0.92	0.86	0.84	0.85	0.99	0.67	0.80

Table F.3: CNN hyperparameters grid search results of the bi-LSTM-CNN-CRF model with stemmed tokens

G

Project plan

Project approach and management plan

Krijn van der Burg

8th April 2020
Version 1.1

Abstract

This Project Approach and Management Plan (PaMP) provides the general framework, strategies and milestones for the project's execution. This is a living document, based on input from the project team and key stakeholders this document's contents may be updated.This project aims to improve the current entity extraction application. Developing a learning based system which can automatically learn to recognize new edge-cases.

Contents

1 Project details

The organisation works with a lot of unstructured data, usually (large) text documents. To understand and bring structure to the document, actors can enter search queries in a (React[1]) web tool. Search queries can consists of words but also entities, e.g. personal identifiers, mobile phone number and URI's. The frontend web tool connects to a (Java[2]) back-end entity extraction service.

1.1 Problem definition

Frequently, actors report entities which weren't found in the given documents by the Java back-end entity extraction service. The entities are not registered named entities under MUC7 and can be considered unique identifiers in both textual and numerical form. E.g. phone numbers, email addresses and URI's. The main problem is identified as the very different ways that certain entities are documented; e.g. phone numbers with and without country codes. These edge cases are increasingly difficult to capture. Rule bases solutions are inherently very strict on the format which these entities do not conform to.

A possible problem to overcome is understanding and properly embedding numerical sequences and unique identifiers as entities. There is no semantic linguistic to understand of said entities. The main key features will probably be the surrounding context and the format of the entity.

1.2 Goal

This project aims to find and create an automated solution to identify all different ways each type of entity is documented. Machine learning is the proposed solution. A Natural Language Processing (NLP) algorithm can be trained on all labelled entities which were found over time. As new entities are found which are not recognized by the machine learning model, e.g. a new zip code format has been introduced, the algorithm can be re-trained to find the new entities. Actors can highlight and label entities which were not found by the model in the same system. By developing a system which periodically re-trains the model, the entity extraction service can automatically be kept up to date.

The promising solution is taking an existing state-of-the-art Named Entity Recognition model and use transfer learning to have it recognize the unique identifier entities as well. The ideal solution is where actors highlight the new labels which weren't found by the model so that it can retrain periodically and also detect the labelled edge cases without any actor involvement.

1.3 Research aspect

The project focuses around finding the optimal solution to a specified problem, meaning the project is research driven. Determining a research question is an essential element to a

[1] `https://reactjs.org/`
[2] `https://java.com/`

qualitative research. Spending time determining research questions will steer the project and create clarity and focus of the deliverables. Both triangulation and laddering were applied to determine the following research questions:

Main research question *How can a machine learning model be realised which identifies entities from text and supports feedback loops?*

Sub questions

- Which annotation scheme best capture an named entity in a document.
- Do token casing features improve NER accuracy?
- Which form of embedding best capture the named entities and its context?
- Which machine learning algorithm and architecture best recognise and classify annotated entities.
- How to compare a production solution and the new developed machine learning model?
- How to automatically deploy a trained model to a production environment?
- Can a ML model size be reduced without significantly compromising its accuracy?

1.3.1 Development oriented triangulation

To reduce common method bias the research questions were formed via triangulation. Approaching the project from different angles and perspective and making sure data and results are cross-checked via different methods.

Library research is done to explore what is already done and what guidelines and theories exist that could help you further your design. Since the advent of the internet library research is also called desk research.

> Answering research questions start by consulting online options and solutions. To determine all possibilities multiple different sources are consulted; including online articles, published articles, and team members expertise.

Field research is done to explore the application context. You apply a field strategy to get to know your end users, their needs, desires and limitations as organizational and physical contexts in which they will use your product.

> To gain a better understanding of the project, set clear software requirements and goals, stakeholders were interviewed and team members were consulted. Using laddering the requirements and research questions were established, covering all aspects of the project.

Lab research is done to test parts or concepts of your product, of the final product. You use lab research to learn if things work out the way you intended them, or to test different scenarios.

> After a first, reliable, prototype is made, the machine learning model will be implemented in a test environment for actors to test. Besides testing the reliability of the system, actors can also provide feedback and criticism. This can be taken into account during the final stages of the realization phase, or be documented in the transfer phase for future improvement.

Showroom research is done to test your ideas in relation to existing work. Showing your prototype to experts can be a form of showroom research or spelling out how your product is different from the competition. Also testing your product to general guidelines is a form of showroom research.

> At the end of the realization phase a presentation will be given, showcasing the prototype and all the achieved results. This is the open centralized moment for feedback from the entire team. The prototype will be compared to the existing solution(s) and all its aspects are compared. When the project moves into the transfer phase, a presentation will be given how the product is intended and to be used, including all the corresponding documentation.

Workshop research is done to explore opportunities. Prototyping, designing and co-creation activities are all ways to gain insights in what is possible and how things could work.

> After defining the project and researching all the possibilities, the proposed solution is discussed with the team. A discussion or brainstorming session will most likely introduce improvements or address new additional points of failure for the project and its proposed solution. Setting the constraints surrounding the prototype together with the team creates a solid foundation for a successful project.

1.3.2 Laddering

To determine the root problem cause the technique laddering is applied to formulate accurate and proper research questions. Laddering works by having the interviewer rephrase the received responses as new questions. A similar technique is called the *5 why's*, where the interviewer subsequently asks *why* five times on the original question to determine the true root reason.

Using these techniques the root cause for this project was uncovered. Which is reducing the amount of manual work to maintain, create and update the regular expressions used to the capture the entities.

Project management

2 Communication

Communication is key in project management. For a successful project execution, effective communication to all stakeholders is essential. Communication is best defined as the exchange of information, though the expression of ideas is just as an important aspect.

2.1 Internal organisation

The project has a centralized weekly meeting to discuss progress and problems which could delay or halt project activities. The entire team has a daily stand-up to discuss work, progress and problems. The decided means of direct online communication is email, but only if communication in person or during meetings cannot take place. Following the 'open-door policy' team members are expected to be open to questions most of the time.

2.2 University

Every Friday a weekly log update is communicated to the internship first assessor. The log contains multiple standard questions to be answered or covered are:

- What did you do last week, and how does that fit into your overall planning?
- What decisions did you take?
- What are you going to do next week?
- What feedback have you received?
- Which problems or issues did you encounter, and if applicable, where do you need help?

3 Resources

All materials, equipment, facilities and employees required for the completion of the project are considered resources.

3.1 Workspace

The organisation provides a physical workplace and the required systems to document results and process data. To communicate with team members an intranet (email) address is provided.

3.2 Data

There is no data available internally. Online open data sources will be consulted to build a local corpus / dataset. As the project is natural language processing focused, the dataset subsequently must be textual. The data must conform two main requirements. Each entity the extraction application is designed to extract must have a relevant context surrounding it. And the entities must be annotated, the word location and its type must be specified. The supplied data are textual documents stored on the local system workspace. Data was provided by the team functional manager.

3.3 Team

A team may consist of sub-teams, groups, and members working solo, e.g. research. Stakeholders are not considered part of the project's resources. The lack of a resource will therefore be considered a risk and constraint on the completion of the respective project activity.

Team role	**Nr #**	**responsibilities**
Product owner	1	Defines Stories and prioritizing the Team Backlog to streamline the execution of program priorities while maintaining the conceptual and technical integrity of the Features or components for the team.
Functional manager	1	Manages and owns the resources of the department and generally directs the technical work of individuals from that functional area who are working on the project.
Scrum master	1	Facilitator for an agile development team. Managing the process how information is exchanged.
Developers	8	Applies the principles of software engineering to the design, development, maintenance, testing, and evaluation of computer software

Table 1: Available human resource

The functional manager and one developer are available nearly every day with a standard meeting scheduled before every sprint to discuss the results and next steps. All of the team members are available for consult. One team member is structurally available for substantive questions and help.

4 Time

The project time-frame ranges from 10-02-2020 till 27-06-2020. Tasks and milestones are organized on a week basis, this time includes the required extensive documentation for the graduation thesis. See Gantt chart 1 for the project time schedule.

4.1 Project phasing

This project uses an agile approach, dividing the project into structured periods of 2 weeks. Tasks are discussed and assigned at the start of each period. At the end of the sprints there will be a demonstration of the products that were made during these sprints.

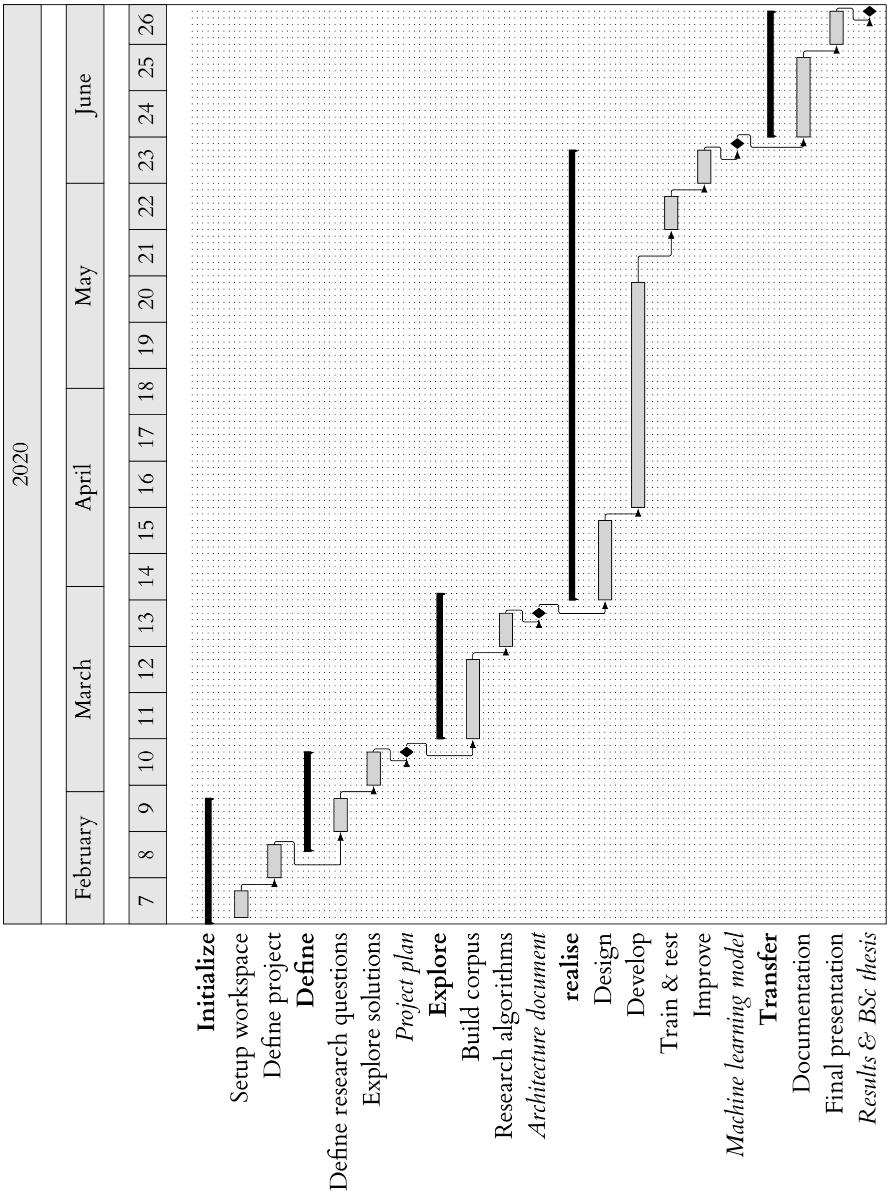


Figure 1: Project Gantt chart time schedule

After that, the plan for the next sprint is made. The progress is documented in a 'live document'.

4.1.1 Initializing

Discussing proposed project along with the available resources into concluding and completing said project. Collecting all necessary information to set the definitions for the following phase.

4.1.2 Defining

During the start of the project, the goal of this phase is to define the project on a broad level. This phase begins with a business proposal. Researching whether the project is feasible and if it should be undertaken. A project plan is created and presented to the stakeholders. Through laddering and triangulation the primary research questions are turned into sub questions to guarantee everything is covered as well as improve the quality of the research.

4.1.3 Exploration

After researching the various available data sources, methods, and solutions to realise the project, a plan of approach is made regarding both the method of analysis methods and software architecture. Including a list of deliverables and all software & research requirements as well as non-functional requirements. Feasibility testing will be performed and completed during this phase, before the development phases start. The plan of approach document must be presented and approved by stakeholders. This phase also includes a research part where the required technologies are compared to determine the best approach. Development environments are also setup, such as installing integrated development environments, continuous integrations and a virtual testing environment

4.1.4 Realization

This is the phase where research questions are answered, and deliverables are developed and completed. Here all previously established functional requirements, non-functional requirements and conclusions from research questions must be included in the product.

4.1.5 Transfer

During the transfer phase; the tools, techniques and methodologies and the team responsible for a succesful transition is identified. Documentation is finished and the final version is presented to the stakeholders.

5 Scope

Defining features and functions of the product and the scope of work needed to complete the project. Although the scope is defined at the start of the project, as the project is continuous it might change depending on stakeholder requirements.

5.1 Use case

For each *action*, an actor can take or interact with, an use case is created. A use case is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal.

Name	Entity extraction
Summary	Actor searches for entity in text
Actor	Registered user
Assumption	Actor is registered and submitted a text or document.
Description	Actor enters and submits search criteria (keyword terms) System applies ML model and displays all results items.
Exceptions	[1] No entities are found : depict error on client UI.
Reults	Actor is shown extracted entities.

5.2 Requirements

The MoSCoW method is a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance placed on the delivery of each requirement. MoSCoW is often used with time boxing, where a deadline is fixed so that the focus must be on the most important requirements.

5.2.1 Must have

Requirements labelled as Must have are critical to the current delivery time box for it to be a success. Requirements can be downgraded from Must have, by agreement with all relevant stakeholders.

ID	Description	MSCW
M_document_1	Documentation that defines the ML model architecture, training process, and hyper parameter tuning.	Must
M_model_1	Machine Learning model that can extract specified entities from text.	Must
M_software_1	Pre-processing the data via an automatic script.	Must

5.2.2 Should have

Requirements labelled as Should have are important but not necessary for delivery in the current delivery time box. While Should have requirements can be as important as Must have, they are often not as time-critical or there may be another way to satisfy the requirement, so that it can be held back until a future delivery time box.

ID	Description	MSCW
S_model_1	Apply grid search or manually explore all hyper parameters.	Should
S_software_1	Create data structure required for training data via an automatic script.	Should
S_software_2	Train and deploy the ML model automatically on a time basis.	Should

5.2.3 Could have

Requirements labelled as Could have are desirable but not necessary and could improve user experience or customer satisfaction for little development cost. These will be included if time and resources permit.

ID	Description	MSCW
C_document_1	Document details and design architecture for real time entity extraction.	Could
C_document_1	(Near) real-time performance metrics update to role-based)	Could

5.2.4 Won't have

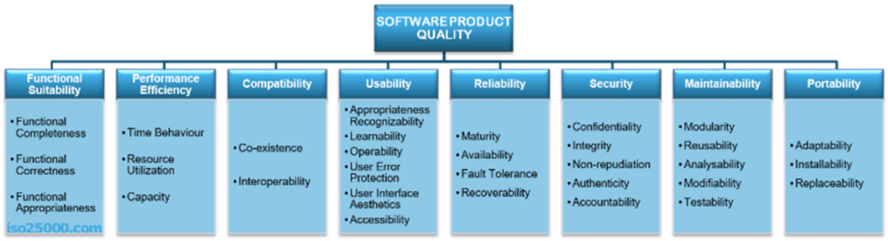
Requirements labelled as Won't have have been agreed by stakeholders as the least-critical, lowest-payback items, or not appropriate at that time. As a result, wont have requirements are not planned into the schedule for the next delivery time box. Won't have requirements are either dropped or reconsidered for inclusion in a later time box.

ID	Description	MSCW
W_model_1	Real-time learning model.	Won't

5.3 Non-functional requirements

System attributes such as security, reliability, performance, maintainability, scalability, and usability are considered and documented as non-functional requirements (NFR). Serving as constraints or restrictions on the design of the system across the different backlogs. Also known as system qualities, NFRs are just as critical as functional Epics, Capabilities, Features, and user Stories. Ensuring the usability and efficacy of the entire system.

NFRs are persistent qualities and constraints that, unlike functional requirements, are typically revisited at every development stage. NFRs are also known as *quality attributes*.



5.3.1 Functional suitability

Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. Composed of the following sub characteristics.

- **Functional completeness.** Degree to which the set of functions covers all the specified tasks and user objectives.
- **Functional correctness.** Degree to which a product or system provides the correct results with the needed degree of precision.
- **Functional appropriateness.** Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

NFR ID	ISO 25010	Description
NFR functional 1	Suitability	Model can extract and identify entities which weren't identified by the current implemented solution.
NFR functional 2	Suitability	Model entity extraction and identification accuracy is determined by a Bayesian metric for evaluation.

5.3.2 Performance efficiency

Performance relative to the amount of resources used under stated conditions. Composed of the following sub characteristics:

- **Time behaviour.** Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
- **Resource utilization.** Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

- **Capacity.** Degree to which the maximum limits of a product or system parameter meet requirements.

NFR ID	ISO 25010	Description
NFR performance 1	Performance	ML model must extract all entities within the given text within a reasonable time relative to text size
NFR performance 2	Performance	Pre-processing must complete within a reasonable time relative to data size.
NFR performance 3	Performance	Model training must complete within a reasonable time relative to training sample size.

5.3.3 Compatibility

Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment. This characteristic is composed of the following sub characteristics:

- **Co-existence.** Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
- **Interoperability**. Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

NFR ID	ISO 25010	Description
NFR compatibility 1	Compatibility	Software may not be operating system nor platform dependant.

5.3.4 Usability

Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This characteristic is composed of the following sub characteristics:

- **Appropriateness recognizability**. Degree to which users can recognize whether a product or system is appropriate for their needs.
- **Learnability**. Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

- **Operability**. Degree to which a product or system has attributes that make it easy to operate and control.
- **User error protection.** Degree to which a system protects users against making errors.
- **User interface aesthetics.** Degree to which a user interface enables pleasing and satisfying interaction for the user.
- **Accessibility.** Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

NFR ID	ISO 25010	Description
NFR usability 1	Usability	Entity extraction results must be easily understandable.
NFR usability 1	Usability	Actor can highlight entities in the frontend which weren't found by the system for feedback learning.

5.3.5 Reliability

System, product or component performing specified functions under specified conditions for a specified period.

- **Maturity.** Degree to which a system, product or component meets needs for reliability under normal operation.
- **Availability.** Degree to which a system, product or component is operational and accessible when required for use.
- **Fault tolerance.** Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
- **Recoverability.** Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

NFR ID	ISO 25010	Description
NFR reliability 1	Reliability	Errors must be graciously handled and prompted to the user.

5.3.6 Security

Product or system protecting information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. Composed of the following sub characteristics:

- **Confidentiality.** Degree to which a product or system ensures that data are accessible only to those authorized to have access.
- **Integrity.** Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data
- **on-repudiation.** Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
- **Accountability.** Degree to which the actions of an entity can be traced uniquely to the entity.
- **Authenticity.** Degree to which the identity of a subject or resource can be proved to be the one claimed.

NFR ID	ISO 25010	Description
NFR security 1	Security	Processing data shouldn't be accessible by other running processes.

5.3.7 Maintainability

Effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. Composed of the following sub characteristics:

- **Modularity.** Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
- **Reusability.** Degree to which an asset can be used in more than one system, or in building other assets.
- **Analyzability.** Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
- **Modifiability.** Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
- **Testability.** Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

5.3.8 Portability

Effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub characteristics:

NFR ID	ISO 25010	Description
NFR maintain 1	Maintainability	(Pre-)processing and model training scripts are bug proof by testing as detailed under the *testing* chapter.
NFR maintain 2	Maintainability	Log preprocessing steps to create a state crumb trail.
NFR maintain 3	Maintainability	Layers can be changed and re-trained without losing trained embeddings of remaining layers.
NFR maintain 4	Maintainability	Descriptive and explanatory readmes and documentation is written.
NFR maintain 5	Maintainability	Source code is algorithmically structured so that future redesign isn't halted.

- **Adaptability.** Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
- **Installability.** Effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
- **Replaceability.** Degree to which a product can replace another specified software product for the same purpose in the same environment.

NFR ID	ISO 25010	Description
NFR portability 1	Portability	ML model must not be language or system dependant.

5.4 Deliverables

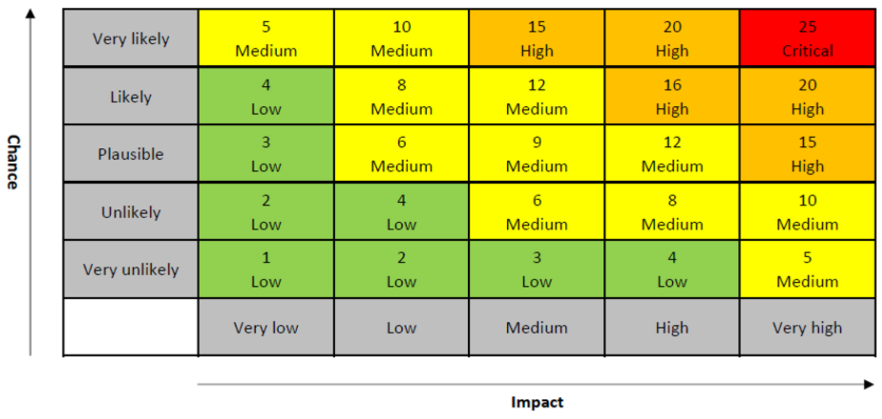
Each deliverable for this project is listed in table 2. A deliverable is a tangible or intangible good, produced service, or developed product which resulted from the project including the corresponding documentation. Deliverables are presented as is when its first presentable version is reached. For an agile project its expected that only metrics and results are delivered during the final sprint, all other deliverables should have been transferred earlier during the project's time frame.

6 Risk

Possible risks which can affect the project, based on the risk matrix table 2. The risk ranges from 1 to 25, where 1 is low and 25 is high. The matrix evaluates the combined occurance chance and impact to determine the risk factor. See figure 2 for a visualization.

Phase	Sprint	Week	Deliverable
Initialize	8	10	Project plan
Explore	4	13	Project architecture document
realise	7	20	Machine learning model
Transfer	10	25	Research results document
Transfer	10	25	Presentation
Transfer	10	25	BSc thesis

Table 2: Deliverables


Chance
Impact

Very likely	5 Medium	10 Medium	15 High	20 High	25 Critical
Likely	4 Low	8 Medium	12 Medium	16 High	20 High
Plausible	3 Low	6 Medium	9 Medium	12 Medium	15 High
Unlikely	2 Low	4 Low	6 Medium	8 Medium	10 Medium
Very unlikely	1 Low	2 Low	3 Low	4 Low	5 Medium
	Very low	Low	Medium	High	Very high

Figure 2: Risk matrix

7 Quality

Project quality is defined as satisfactory when the resulting product or service is deemed positively suitable for all its intended purposes. More specifically when all systemic requirements, both functional and non-functional, have been implemented and its result tested.

7.1 Testing

Goal of this testplan chapter is to describe the test approach. In this document each person or party concered is informed of all various testing approaches, its activities and the products to deliver. This plan describes the following types of tests.

> The project is designed to catch, extract, and identify entities which the existing solution couldn't. The exact purpose of the project is to cover all the edge-cases. which includes testing all entities that weren't found by the existing solution.

Description	Recurrence chance	Impact	Risk score
Lack of knowledge	Very likely	Low	10 Medium
Poor quality data sources	Probable	Medium	9 Medium
Not enough information available	Unlikely	Medium	6 Medium
Unable to give advice from our information	Unlikely	High	8 Medium
Loss of group member(-s)	Likely	Medium	12 Medium
Loss of data	Very unlikely	Very high	5 Medium
Lack of functional tooling	unlikely	medium	6 Medium
Receive the data late	Likely	High	16 High

Table 3: Risk matrix

Description	Prevention	Mitigation
Lack of knowledge	Exchange information between group members	Learning from tutorials, teachers and other resources
Poor quality data sources	Consider multiple possible sources before selecting.	Determine alternative sources.
Lack of information	Use multiple differente sources, communicate with product owner.	Research higher quality sources.
No possible advice	Have a good insight of the data.	Spend more time to look further and deeper into the data for more insight.
Loss of team member(-s)	Maintain good communication between group members.	Redistribute the lost group members task. Discuss if requirements should be changed.
Loss of data	Have multiple backups of the data in different locations.	Request another copy of the data from the product owner.
Lack of functional tooling	Unable to efficiently analyse and work with data.	Look for different tools, or go for a less efficient solution.
Receive the data late	Contact data source early on, remind them if needed.	Use (open source) dummy data in the same format.

Table 4: Risk prevention and mitigation strategy

7.1.1 Unit tests

Testing specific critical methods within the application. Comparing the output of a method to a given value and determine if its the correct output. Only used to test individual methods that have some kind of output whether it be implemented in the method or on the file system.

7.1.2 System tests

Testing the described functionalities use cases defined in the User Requirements Specifications (URS) chapter. Each test is defined by whether the use case can be achieved ad described without any limitations, bugs or quirks. If the functionality can be executed in full as described and meet standard expectations of the user, the test is accepted.

7.1.3 Component tests

Testing system components which are reliable on other components. The component thats being tested uses or accesses mock components that it normally would access. This way each component can be tested under different circumstances as the mock components can be designed to fit any situation and circumstance, given the appropriate values.

7.1.4 Model quality

To assure the quality of the machine learning model, the training process uses various features to prevent under- and over-fitting. These methods include *synonym replacement*, *random word replacement*, *random word removal*, and *random word swap* to provide different contexts and introducing random noise.

8 Cost

All parts of the project which have an associated cost are mentioned here. Costs range from monetary to an employee's available work time. By pre-allocating costs for the project possible future delays are prevented. Project activities can be delayed or even completely halted when resources aren't available or cannot be provided due to an incorrect cost estimation.

> For this project there are no relevant costs identified. Employee cost is determined insignificant due to the salary scale. Required data processing can be performed on the personal workspace systems of each employee.

8.1 Procurement

If the hardware is deemed incapable of processing the data in an acceptable time, there are other more powerful systems available.