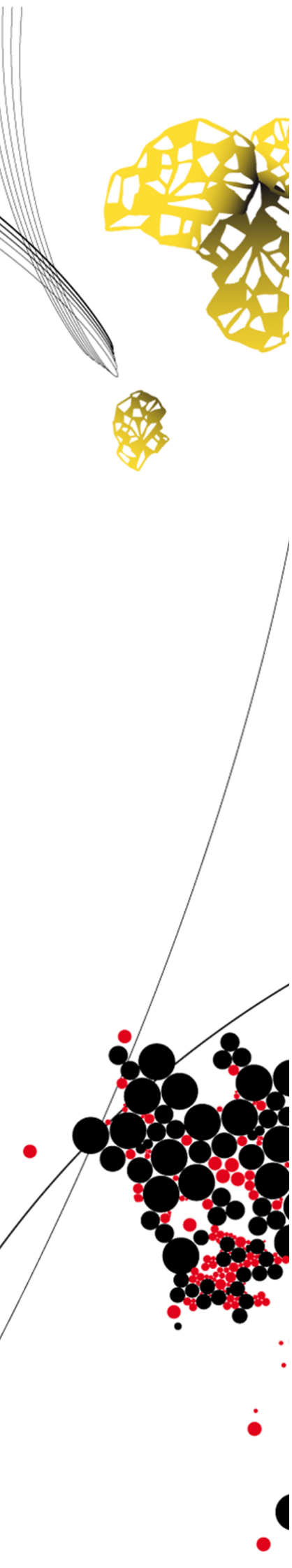


UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

Attribute Extraction from Darknet Markets and the Applicability of Transfer Learning

M.Sc. Thesis by
Nicole Oonk

Master in Computer Science
Specialization in Data Science and Technology

November 2018

Graduation Committee:
University of Twente
dr. C. Seifert
dr. D. Bucur

Team High Tech Crime
dr.ir. A.H. van Bunningen

Datamanagement and Biometrics Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Cybercrime is an international threat that is continuously growing. One of the fundamental strategies in the combat against cybercrime is intelligence-led policing. Darknet markets (DNMs) are a rich source of information on cybercrime. However, gaining in depth insight into the products and services offered on DNMs is difficult. Whereas studies have been conducted to gain insight into DNMs, no study has yet focussed on extracting information from DNM posts to gain insight in the techniques that are exploited, the locations that are mentioned and other characteristics law enforcement agencies are interested in.

In this study, attribute extraction was performed to automatically extract attributes from bulletproof hosting advertisements of a multi-lingual DNM. A variety of attribute extraction techniques was tested and compared. The extraction techniques included are dictionary-based, rule- and pattern-based and machine learning based attribute extraction. The machine learning algorithms that were compared are Maximum Entropy Model (MEM), Conditional Random Field (CRF) and bidirectional Long-Short Term Memory with Conditional Random Field (biLSTM-CRF). Furthermore, transfer learning by fine-tuning was applied to improve the performance of the biLSTM-CRF extractor.

Russian attribute extraction was performed with a micro-averaged F2-score of 0.886, using the biLSTM-CRF extractor developed with transfer learning. The ensemble extractor, that combines for each attribute the best extraction technique, resulted in a micro-averaged F2-score of 0.859 for English attribute extraction. Transfer learning applied to the biLSTM-CRF extractor resulted in a performance improvement of 0.014 and 0.002 micro-averaged F2-score for Russian and English respectively. With attributes that improved with up to 0.074 F2-score.

These results show that attribute extraction and transfer learning have a great potential to gain insight in DNMs and effectively include DNMs in intelligence-led policing.

Acknowledgements

This master's thesis marks the end of my five-year period as a student at the University of Twente. The last several months of carrying out my graduation project have been a great experience for me. I would like to take this opportunity to thank several persons that helped make this project possible. First of all, I would like to thank my supervisors Christin Seifert, Doina Bucur and external supervisor Arthur van Bunningen for taking the time to guide me during this project with meetings to discuss the work and valuable feedback to improve this thesis. Our meetings always left me feeling motivated and full of ideas. Furthermore, I would like to express my gratitude to all my colleagues for being able to conduct this project at Team High Tech Crime of the Dutch National Police. Lastly, I would like to thank my family and friends for their continuous support during this project.

Contents

Abstract iii

Acknowledgements v

List of Acronyms xiii

1 Introduction 1
- 1.1 Problem Statement . . . 2
- 1.2 Research Goal and Questions . . . 2
- 1.3 Report Structure . . . 3

2 Domain Background Information 5
- 2.1 Layers of the World Wide Web . . . 5
- 2.2 Bulletproof Hosting on Darknet Markets . . . 6

3 Related Work 9
- 3.1 Attribute Extraction . . . 9
  - 3.1.1 Dictionary Based Approach . . . 9
  - 3.1.2 Rule and Pattern Based Approach . . . 10
  - 3.1.3 Machine Learning Approach . . . 10
  - 3.1.4 Suitability of Extraction Techniques . . . 11
- 3.2 Transfer Learning for Attribute Extraction . . . 13
  - 3.2.1 Traditional Machine Learning . . . 13
  - 3.2.2 Neural Networks . . . 13
- 3.3 Attribute Extraction and Transfer Learning on DNMs . . . 14
  - 3.3.1 DNM Attribute Extraction . . . 14
  - 3.3.2 Transfer Learning for DNM Attribute Extraction . . . 15
- 3.4 Summary . . . 15

4 Theoretical Background 17
- 4.1 Supervised Classification with Machine Learning . . . 17
  - 4.1.1 Supervised Classification . . . 17
  - 4.1.2 Sequence Labelling . . . 18
  - 4.1.3 Maximum Entropy Model . . . 19
  - 4.1.4 Conditional Random Field . . . 19
  - 4.1.5 Long Short-Term Memory Neural Network . . . 21
- 4.2 Transfer Learning . . . 24
  - 4.2.1 Notations and Definitions . . . 24
  - 4.2.2 Transfer Learning Categories . . . 24
- 4.3 Evaluation . . . 25
  - 4.3.1 Performance Measures . . . 26
- 4.4 Summary . . . 27

5 Dataset **29**
5.1 Dataset Construction . . . 29
5.2 Exploratory Data Analysis . . . 30
5.3 Data Preparation . . . 33
5.3.1 Automatic Split of English and Russian Posts . . . 34
5.3.2 Final Russian and English Dataset . . . 34
5.4 Attribute Identification . . . 35
5.5 Manual Annotation . . . 37

6 Methodology **39**
6.1 Baseline Extractors . . . 40
6.2 Dictionary Extractors . . . 41
6.3 Rule and Pattern Extractors . . . 42
6.4 CRF Extractors . . . 44
6.4.1 Initial CRF Extractor . . . 44
6.4.2 Experiments . . . 45
6.5 Hybrid CRF Extractors . . . 46
6.6 biLSTM-CRF Extractors . . . 47
6.6.1 Initial biLSTM-CRF . . . 47
6.6.2 Experiments . . . 48
6.7 Transfer Learning with the biLSTM-CRF Extractors . . . 49
6.7.1 Experiment 1: Transferring Word Embeddings vs. Word and Character Embeddings . . . 49
6.7.2 Experiment 2: Comparing Source Datasets . . . 50
6.8 Ensemble Extractors . . . 50
6.9 Training and Testing . . . 50

7 Experiments and Results **51**
7.1 Experimental Setup . . . 51
7.2 Result Overview . . . 52
7.3 Baseline Extractors . . . 54
7.4 Dictionary Extractors . . . 55
7.5 Rule and Pattern Extractors . . . 58
7.6 CRF Extractors . . . 59
7.7 Hybrid CRF Extractors . . . 62
7.8 biLSTM-CRF Extractors . . . 65
7.9 Transfer Learning with the biLSTM-CRF Extractors . . . 68

8 Conclusion and Discussion **71**
8.1 Research Questions . . . 71
8.2 Discussion and Limitations . . . 72
8.3 Future Work . . . 73

A Hyper-parameter Tuning **75**

B Feature Engineering **77**

C Annotation Guidelines **81**

D Baseline Results **87**

Bibliography **91**

List of Figures

1.1 Research problem input and output 3

2.1 Layers of the World Wide Web 6
2.2 Russian BPH advertisement 7
2.3 English BPH advertisement 8

3.1 Applicability of extraction methods depending on text and attribute type 12

4.1 Graphical representation of the Linear Chain CRF [39] 20
4.2 RNN neuron 21
4.3 LSTM network with gated cells 23
4.4 Types of predicting errors 26

5.1 Post lengths of the 844 BPH posts 30
5.2 Post count per Language 31
5.3 Length of English, Russian and all posts with five or more tokens . . . 31
5.4 BPH thread post count 32
5.5 Post length per post index in BPH threads 32
5.6 Post count per day of week 33
5.7 Post count per time of day 33
5.8 Class frequency distribution 37

6.1 Aggregated dictionary created from attribute specific dictionaries . . . 41

List of Tables

4.1 Transfer learning categories 25

5.1 Post, paragraph and sentence count in the Russian and English datasets 35
5.2 Sixteen identified attributes in BPH posts 36

6.1 Extraction techniques used per attribute 39

7.1 Attribution extraction result overview - The best F2-scores 53
7.2 F2-scores of the original baseline extractor and the impact of lowercasing (L) and overlap removal (OR) 54
7.3 $F2_{Mi}$-scores of the aggregated and original baseline extractor 55
7.4 The F2-score (F2), precision (P) and recall (R) for dictionary-based location extraction and the impact of lowercasing (L) and overlap removal (OR) 56
7.5 The F2-score (F2), precision (P) and recall (R) for dictionary-based currency extraction and the impact of lowercasing (L) and overlap removal (OR) 57
7.6 The F2-score (F2), precision (P) and recall (R) for dictionary-based payment period extraction and the impact of overlap removal (OR) . . 57
7.7 The F2-score (F2), precision (P) and recall (R) for rule-based extraction of ICQ, jabber mail, URL and price attributes 58
7.8 Performance scores of the initial CRF extractor and MEM extractor . . 59
7.9 CRF extractor - The performance impact of post, paragraph and sentence sequence definitions 60
7.10 CRF extractor - The performance impact of hyper-parameter tuning . . 60
7.11 CRF extractor - The performance impact of using the initial, all or all beneficial features 61
7.12 CRF extractor - The performance impact of using context features with a window of 1, 3 and 5 61
7.13 CRF extractor - The performance impact of data pre-processing steps . 62
7.14 Final CRF extractor and initial CRF extractor F2-scores 63
7.15 Hybrid CRF extractor - The performance impact of dictionary and rule features 63
7.16 Hybrid CRF extractor - The performance impact of transfer learning features 64
7.17 Final CRF extractor and final hybrid CRF extractor F2-scores 65
7.18 biLSTM-CRF extractor - The performance impact of post, paragraph and sentence sequence definitions 66
7.19 biLSTM-CRF extractor - The performance impact of adapting hyper-parameter values and architecture 67
7.20 biLSTM-CRF extractor - The performance impact of data pre-processing steps 67
7.21 Initial biLSTM-CRF extractor and final biLSTM-CRF extractor F2-scores 68

7.22 Transfer learning - The performance impact of transferring word embedding versus word and character embedding layers 69
7.23 Transfer learning - The performance impact of the source datasets . . . 70

A.1 Hyper-parameter tuning phase 1: Discrete Hyper-parameters with the tested parameter values and the tuned values 75
A.2 Hyper-parameter tuning phase 2: Continuous Hyper-parameters with the tested parameter values and the tuned values 75

B.1 Initial feature set 77
B.2 Extended feature set part 1 78
B.3 Extended feature set part 2 79
B.4 Extended feature set part 3 80

D.1 Baseline performance scores - English 88
D.2 Baseline performance scores - Russian 89

List of Acronyms

BB-code	Bulletin Board Code
biLSTM-CRF	Bidirectional Long-Short Term Memory with Conditional Random Field
BPH	Bulletproof Hosting
CRF	Conditional Random Field
DNMs	Darknet Markets
HMM	Hidden Markov Model
IE	Information Extraction
IOCTA	Internet Organised Crime Threat Assessment
LM	Language Model
LSTM	Long Short-Term Memory
MEM	Maximum Entropy Model
MEMM	Maximum Entropy Markov Model
MUC	Message Understanding Conference
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part Of Speech
RNN	Recurrent Neural Network
SVM	Support Vector Machines
Tor	The Onion Router

Chapter 1

Introduction

Cybercrime is an international threat with extensive consequences. The global cost of cybercrime in 2017 was estimated to be as much as 600 billion US dollars [1]. Furthermore, the threat is continuously growing. Cybercriminals enhance their modus operandi by rapidly exploiting technical innovations and developing protective countermeasures against police operations. Moreover, other criminals are attracted to cybercriminality due to the seemingly low risk of getting caught and the ease of getting started with the availability of guides, tools and crimeware-as-a-service [2].

Cybercrime is vigorously combated by national law enforcement agencies as well as international organizations including Europol. To direct the operational focus of this fight, Europol publishes an annual strategic report known as the Internet Organised Crime Threat Assessment (IOCTA) report. This report presents the key findings, new threats and developments in cybercrime. Fundamental in the strategy of 2018 and preceding years is intelligence-led policing [3]. This includes collecting, processing and analysing information on cybercrime from public and private sources. Furthermore, it was predicted that promising techniques such as machine learning will become invaluable for intelligence-led policing. Therefore, it was strongly recommended to continuously explore opportunities enabled by these techniques.

Darknet Markets (DNMs) are an importance source of information on cybercrime. DNMs are illicit online markets on the dark web that allow criminals to communicate anonymously and participate in a worldwide trade of illicit products and services of which many enable other criminal activities [3]. These markets are intentionally hidden and are only accessible using a browser based on an anonymity network, such as The Onion Router (Tor). This and other measures are used to reduce traceability of DNM users and that presents large limitations for police operations. Studies have been conducted to gather intelligence from and gain insight in DNMs. These include exploring the DNM user migration to other DNMs after Operation Bayonet successfully took down the two prominent DNMs Alphabay and Hansa Market [4], the products and services sold on DNMs [5], the business models used by sellers [2] and methods to increase traceability of DNM users such as authorship analysis [6].

The IOCTA report is based on a set of cybercrime priorities determined by Europol. These priorities include cross-cutting cybercrime enablers [3], i.e. activities that deal with more than one area of cybercrime, but are not necessarily illegal themselves. Bulletproof Hosting (BPH) is an important enabler that is offered as a service on DNMs. BPH provides hosting of illicit content, such as phishing sites and malware, that enables others to commit crimes.

1.1 Problem Statement

There is an unquestionable need for intelligence to effectively combat cybercrime. DNMs provide a rich source of information about cybercrime including cross-cutting cybercrime enablers, such as BPH. DNMs contain in depth information about BPH, such as the techniques used and the locations of the servers. This knowledge can, for example, be used to discover what technological innovations BPH providers have exploited. To further illustrate the value of DNMs, imagine a fictional scenario where the police is looking for a provider of BPH services that match a specific set of characteristics. Since BPH providers commonly promote their services on DNMs, the content on DNMs can be inspected for BPH advertisements that match these characteristics to potentially find its DNM presence and gather information on the provider.

However, obtaining the subset of DNM content that is useful for investigations or gaining in depth insight into the BPH services offered on DNMs is complicated. These tasks cannot be performed manually due to the vast amount of unstructured, multilingual text data to comb through in combination with limited time, budget and employees. Therefore a method should be developed that automatically recognizes and extracts the information law enforcement is interested in. This can help law enforcement agencies to increase their insight in DNMs and enable intelligence-led policing.

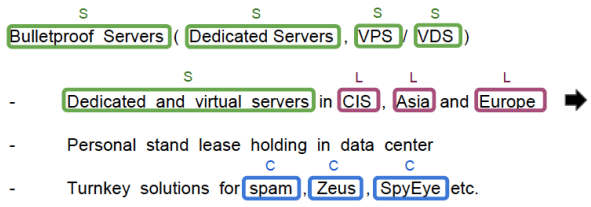
Automatic attribute extraction has been successfully applied to a variety of domains including biomedical attribute extraction [7] and product attribute extraction [8] to extract information. DNMs have not received much attention. In 2017, Durrett et al. conducted an initial attempt at attribute extraction for DNMs in general [9][10]. In their studies, they developed attribute extractors for product and price information using Support Vector Machines (SVM). However, attribute extraction to gain in depth information about specific topics including BPH has not been studied yet.

1.2 Research Goal and Questions

In this research, an initial attempt at attribute extraction from BPH advertisements or post on DNMs was conducted as a step towards effectively including DNMs in intelligence-led policing. For this attempt, we focussed on the BPH posts of one DNM. The goal of this research is to develop attribute extractors to automatically extract attributes relevant for police investigations from these BPH posts. This is visualized in figure 1.1. Examples of relevant attributes are the type of service, technical specifications and contact details of the BPH provider.

For this application, the task of finding all posts, and therefore attributes, relevant for an investigation is of importance. Therefore, a variety of attribute extraction techniques was tested and compared to find the best performing technique for each attribute. Furthermore, transfer learning was applied in an attempt to improve the extractor performance. Transfer learning is a technique to reuse knowledge learned by a pre-trained model for a new model, such as our attribute extractors. In other words, transferring knowledge from the old to the new model. As shown in previous research [11] [7], using pre-trained models can greatly reduce the error rate for language processing tasks. Moreover, models can be trained on significantly less

labelled data, reducing the common bottleneck of labelling natural text for attribute extraction.



Attribute	Attribute instances
Service (S)	Bulletproof Servers, Dedicated Servers, Dedicated and virtual servers, VDS, VPS
Location (L)	Asia, CIS, Europe
Content (C)	spam, SpyEye, Zeus

FIGURE 1.1: BPH posts as problem input (left) and extracted attributes as problem output (right)

The research goal described above resulted in the following research questions:

RQ1: Which attributes are present in DNM posts advertising bulletproof hosting services?
After exploring the dataset, generic and BPH specific attributes in BPH posts that are relevant for law enforcement operations were identified. For this purpose, a subset of the data was manually inspected. The identified attributes were then verified by a domain expert.

RQ2: How well can these attributes be extracted from DNM posts advertising bulletproof hosting services?
Attribute extractors were developed to automatically extract these attributes from BPH posts. Several techniques were compared. These are dictionary-based attribute extraction, rule- and pattern-based attribute extraction and machine learning based attribute extraction. The machine learning algorithms that were compared are Maximum Entropy Model (MEM), Conditional Random Field (CRF) and Bidirectional Long-Short Term Memory with Conditional Random Field (biLSTM-CRF). Furthermore, for each technique, experiments were conducted to improve the performance of the attribute extractors. By combining the best performing technique of each attribute, an ensemble model was created.

RQ3: What is the impact of applying transfer learning on attribute extraction from DNM posts advertising bulletproof hosting services?
After developing the attribute extractors for research question 2, the potential of using transfer learning to improve the machine learning based extractors was explored. Transfer learning was applied to the CRF, a traditional machine learning model, by enriching the input data with the output of a model that was pre-trained on a large corpus. Transfer learning was applied to the biLSTM-CRF, a neural network, by pre-training the embedding layer of the model on a large corpus. Furthermore, experiments were conducted to test the impact of the corpus used for pre-training.

1.3 Report Structure

The structure of the remainder of this report is as follows: Chapter 2 provides background information on the dark web, darknet markets and bulletproof hosting. Chapter 3 presents related work about attribute extraction and transfer learning and describes how these fields have been applied in existing studies about DNMs. Chapter

4 provides theoretical background of the techniques used in this study. Chapter 5 explores the dataset and describes the data preparation, attribute identification and manual annotation processes. Chapter 6 describes the research methodology that was used to answer the research questions. In chapter 7, the results are presented and evaluated. Finally, in chapter 8, a conclusion is drawn followed by a discussion of the work and promising steps for future work.

Chapter 2

Domain Background Information

In this chapter domain background information is provided. First, the dark web and the position of the dark web within the complete World Wide Web is explained including the main reason criminals are attracted to the dark web to perform illicit practices. Afterwards, bulletproof hosting and darknet markets are explained for insight into bulletproof hosting services and how darknet markets are used to sell them.

2.1 Layers of the World Wide Web

The World Wide Web can be divided in three main parts: the surface, deep and dark web. Web pages belong to one of these parts based on their accessibility and whether or not the pages are indexed by standard search engines, such as Google. The dark web covers a small portion of the web. However, since the majority of the web is hidden, the actual proportions can only be estimated. The difficulty of estimating the proportions of the web is emphasised by the resulting estimated proportions of the surface web. The surface web has been estimated to cover almost ten percent [12] whereas others assign less that one percent of the web pages to the surface web [13][14]. Figure 2.1[1] visualizes the three layers of the web including their estimated proportions.

Web pages of the surface web are indexed by search engines and accessible to anyone with an Internet connection. Therefore, all search results that are returned by search engines are part of the surface web. The deep web consists of web pages that are not indexed and commonly require a password to be accessed. Examples are online banking accounts and the content of Dropbox accounts. The deep web covers the majority of the web.

The dark web is an encrypted network that is often used for illicit purposes. It is deliberately kept hidden and therefore the web pages are not indexed and cannot be accessed with a regular browser. Commonly, a Tor browser is required to access the encrypted network and the dark web pages. The encrypted network and Tor browser result in deep layers of anonymity for both hosters and users. The level of anonymity is the main reason criminals are attracted to the dark web [15]. With the assurance of anonymity, criminals are free to perform their illicit practices without having to worry about detection by law enforcement and their online presence being traced back to their real identities.

[1] Source: https://ucsd.libguides.com/CAT3/web-sources

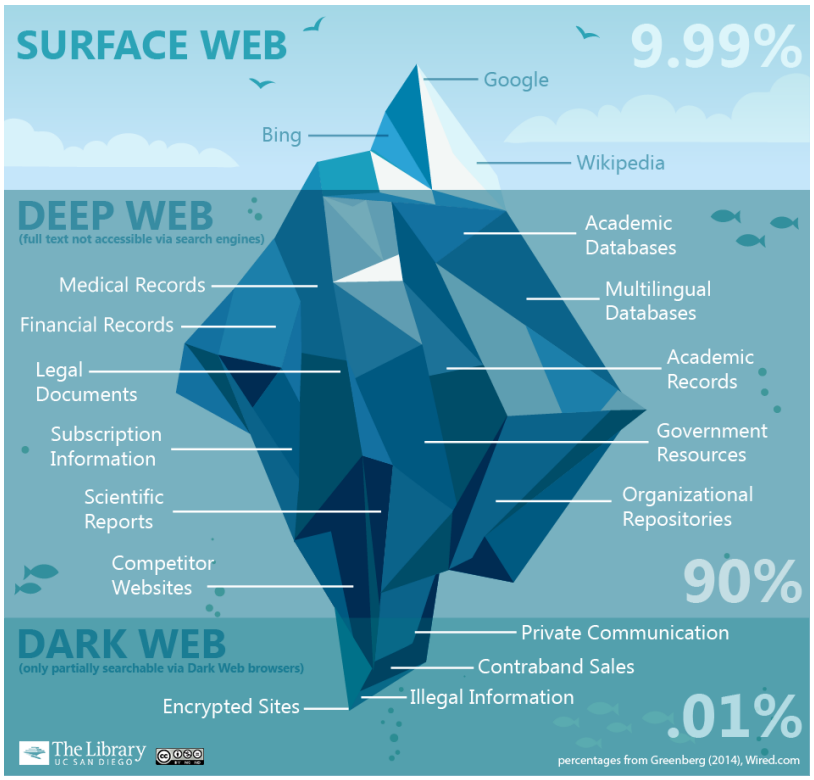


FIGURE 2.1: Layers of the World Wide Web[2]

2.2 Bulletproof Hosting on Darknet Markets

DNMs are located on the dark web and provide a platform for anonymous communication and worldwide sale and purchase of illicit goods and services. With the availability of knowledge, goods and services on DNMs it has become easier to conduct illicit activities. Instead of mastering the technical skills and gathering all resources themselves, criminals can make online purchases of products and services that allow them to realize their plans. This business model is called crimeware-as-a-service [2]. Whilst DNMs exist in varying languages, Russian and English were found to be the predominant languages on many DNMs [16] [17].

The interface of a DNM can be organized as a regular online market, such as eBay, or it can be organized as a forum [3]. For this study we focus on forum style DNMs. The content on these DNMs is commonly stored and presented in a structured way. The structure relevant for this study is as follows: The top level is the DNM itself that is split in categories to store content by topic. In each category, threads can be found that consist of posts. The categories are not split in subtopics. Therefore, threads with related sub topics are shown on the same page without further filtering options. One of these subtopics is bulletproof hosting (BPH).

BPH is a cross-cutting cybercrime enabler that is offered on DNMs. BPH is similar to regular hosting with a hosting party with access to a data center consisting of numerous servers on the one hand and customers that pay to have their content hosted on the other hand. The main difference can be found in the type of content that is allowed to be hosted. Where regular hosting services have strict regulations that

[2]See footnote 1.

forbid malicious and illegal content, bulletproof hosting providers are more lenient in the type of content customers can upload and allow for example darknet markets, phishing sites, malware and botnet command and control centers. Furthermore, besides regular hosting features, such as SSL certificate and Distributed Denial or Service protection, bulletproof hosting offers features specific for the protection of customers against detection by, i.a., law enforcement.

A common method to promote an available BPH service on DNMs is for the provider to start a new conversation, a thread, on the platform and to assign it the category designated for hosting and related services. In the initial post, the service is introduced and an overview is given of all types of the services offered including their characteristics and price. Furthermore, the first post commonly contains the terms of service explaining the content that is and is not allowed, the availability of the provider for support and the contact details to ask questions and purchase the service. Figure 2.2[3] and 2.3[4] show publicly available examples of a Russian and English BPH posts. Noticeable is the occurrence of several English words in the Russian post. In case the BPH provider extends its services through e.g. the purchase of new servers, a new post is commonly added to the existing thread to promote the extension of the service instead of creating a new thread. Details such as terms and conditions and the contact details mentioned in earlier posts apply on the extended service unless otherwise stated. These subsequent posts can be kept short as the details can be found in the initial post. Other options are to create a post similar to the initial post with details being repeated or to use a mixture of the two. Furthermore, a thread can consist of e.g. reviews and questions posted by others and responses of the provider.

Наш сервис предоставляет вам услуги регистрации **Абузоуйстойчивых доменов**.

Уже достаточно продолжительное время мы регистрируем домены под:
- DNS
- Pharma
- Spam (редиректы и основные домены, недорогие варианты под массовую регистрацию и надежные штучные экземпляры)
- Scam
- Phishing (от мелких биллингов до интернет-банков)
- Дроп-проекты
- Трояны в любом виде (в т.ч. антивирусы, кодеки и др.)
- Ботнеты
- etc.

FIGURE 2.2: Russian BPH advertisement[5]- A publicly available BPH advertisement with mixed languages.

[3]Source: https://securelist.com/the-botnet-ecosystem/36279/

[4]Source: https://blog.malwarebytes.com/threat-analysis/2012/11/citadel-a-cyber-criminals-ultimate-weapon/

[5]See footnote 3.

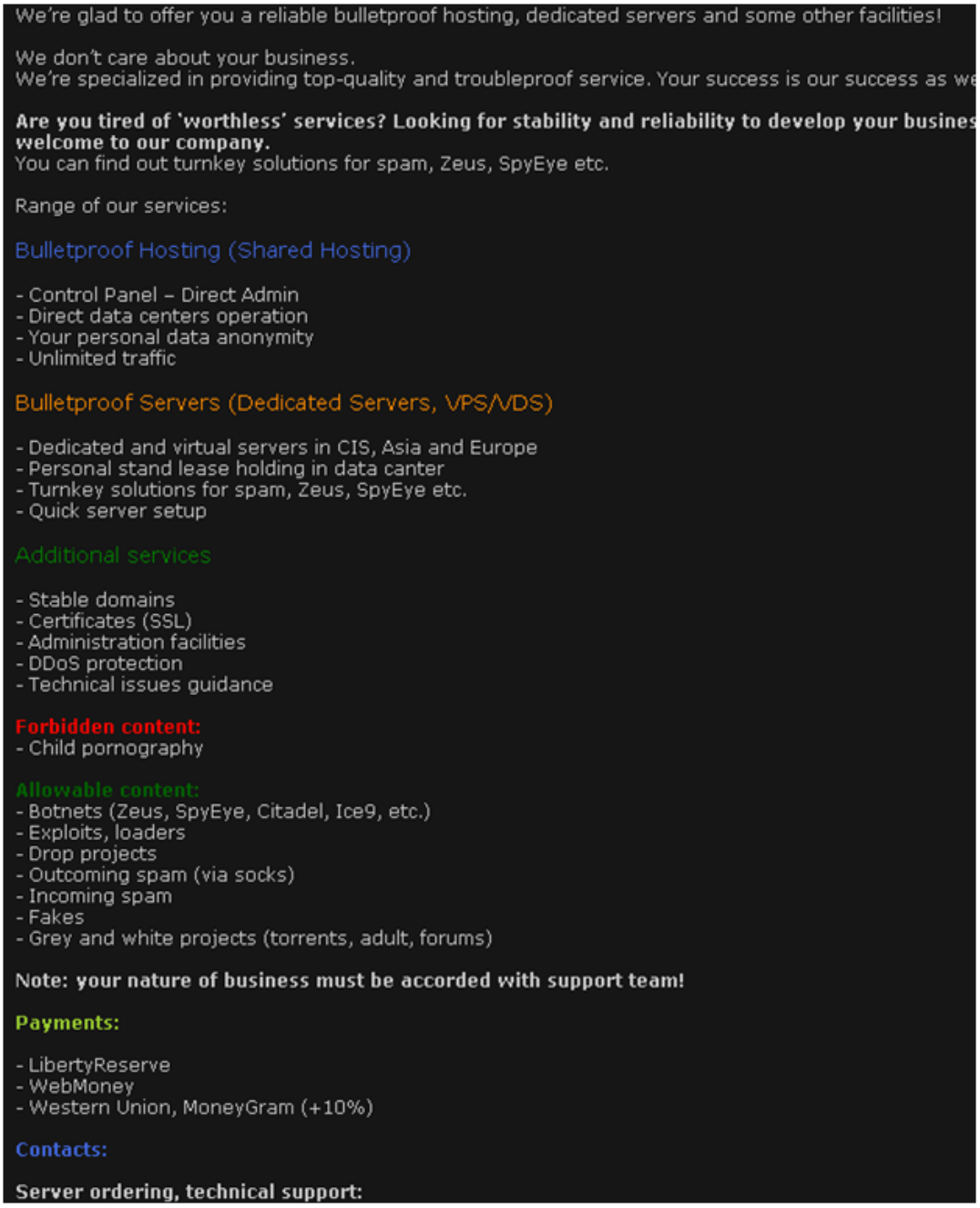


FIGURE 2.3: English BPH advertisement[6]- Despite several cut-off sentences, this publicly available image gives a good example of an English BPH advertisement.

[6]See footnote 4.

Chapter 3

Related Work

This chapter describes how the fields of attribute extraction and transfer learning for attribute extraction have been researched in other studies. Both fields have been applied to a variety of domains. However, DNMs have not received much attention. Therefore, in section 3.1 and 3.2, a domain independent explanation of attribute extraction and the application of transfer learning is provided. Afterwards, in section 3.3, two related studies on DNMs are explored in detail. Finally, section 3.4 provides a summary of the related work and describes how it relates to our study.

3.1 Attribute Extraction

Attribute extraction is the task of automatically discovering and extracting attributes from text. It is a sub field of Information Extraction (IE) and Natural Language Processing (NLP) and is applied on various domains including biomedical attribute extraction [7] and product attribute extraction [8].

For attribute extraction and other sequence labelling tasks such as Named Entity Recognition (NER), three main categories of extraction techniques can be distinguished: (i) dictionaries, (ii) rules and patterns and (iii) machine learning [18][19]. The techniques can be used separately or in combination to make hybrid models. The subsections 3.1.1 to 3.1.3 explain the extraction techniques. Afterwards, section 3.1.4 provides an overview of in what situations the techniques are suitable.

3.1.1 Dictionary Based Approach

The dictionary-based approach makes use of a prepared dictionary or gazetteer to scan for known attributes in the text [8]. The dictionary consists of known attributes, e.g. a currency dictionary containing the known currencies.

For a simple dictionary lookup as described above, a dictionary can be constructed by a domain expert or a publicly available dictionary can be used. However, a common downside is a low precision due to ambiguity of the dictionary values and a low recall when the dictionary does not provide sufficient coverage of the possible attribute values. To increase the performance of dictionary-based attribute extraction, studies have been conducted to, e.g., automatically generate dictionaries using Wikipedia or to combine the dictionary-based and machine learning approach into a hybrid attribute extractor [20].

3.1.2 Rule and Pattern Based Approach

For the rule- and pattern-based approach, linguistic extraction rules or patterns are crafted that are able to locate the attribute based on, e.g., the grammatical structure in the text [18][21]. An example is a regular expression to extract email addresses or credit card numbers.

A downside of this approach is the complex and time-consuming task of crafting rules and patterns that avoid false positives and at the same time cover the possible structures the attribute can take. To overcome this limitation, studies have been conducted to automate this process. For example, Li et al. [22] developed ReLIE, an algorithm to learn a regular expression by updating an initial regular expression. The purpose of this algorithm is to find the set of transformations to an initial regular expression that maximizes the model's F1-score using labelled training data.

3.1.3 Machine Learning Approach

For the machine learning approach, a model is trained to recognize attributes in text using statistical techniques and a training dataset [18][23]. This section describes supervised machine learning and neural networks for attribute extraction.

Supervised Machine Learning:

For supervised attribute extraction, i.e. with supervised machine learning, a model is trained based on examples of labelled texts. Therefore, a training dataset should consist of example texts and labels that indicate the attributes in each text. These labels should indicate both the class and boundary, since attributes can encompass more than one word, such as 'dedicated servers' [24]. A common approach is to use the BIO encoding [18]. BIO stands for Beginning, Inside and Outside of attributes. Using the 'dedicated servers' service attribute as example, the BIO encoding would label the first word as B_service and all following words that are part of the same attribute instance as I_service as shown below.

Example: $[\textbf{dedicated}]_{B\_service}[\textbf{servers}]_{I\_service}[\textbf{and}]_{O}$ ...

To train models based on the labelled training dataset, each word is first represented as a feature vector. This process is called feature engineering. From these feature vectors, patterns can be inferred that enable the model to distinguish between the classes and to determine the boundaries of attributes. The process of feature engineering can result in various types of features. Nadeau et al. [25] divided the feature types in three categories: (i) word-level features, (ii) list lookup features and (iii) document and corpus features. Word-level features provide information about the character makeup of the tokens, e.g., the suffix of the token and whether the token starts with a capital. List lookup features indicate whether the token is in a predefined dictionary. Document and corpus features provide information about the document and corpus, e.g., the token frequency in the corpus. Other features that can be used to boost the performance are character-level embeddings [26] and word representations, such as clustering-based representation and word embeddings [27]. Furthermore, features based on other natural language processing tasks can be included, such as Part Of Speech (POS) and chunk tags [28]. Adding results from other models to the feature vectors to increase the performance of the attribute extractor

is a type of transfer learning that is further elaborated in section 3.2.

Various traditional machine learning algorithms have been used for attribute extraction in previous studies. These include sequence models that classify each word whilst taking into account the word context, e.g. the surrounding words and their predicted labels [18][23][24]. Examples are Hidden Markov Model (HMM), CRF and Maximum Entropy Markov Model (MEMM). Furthermore, algorithms that classify each word separately have been used [23]. Examples are support vector machines, decision trees, boosting and maximum entropy. Whilst these latter algorithms classify each word separately, they can take the context of a word into account by adding the features of the surrounding words to the feature vector.

Unsupervised machine learning techniques such as clustering can be used to overcome the task of data labelling [25]. However, unsupervised machine learning is difficult to evaluate and is often applied in combination with supervised learning to augment the handcrafted features instead of as a standalone model [24][29].

Neural Networks:

Recent work has shown that neural networks can be successfully applied to NLP tasks such as attribute extraction. The biLSTM-CRF, is a combination of models that is commonly used and has proven to work for attribute extraction [30]. Instead of handcrafted features, the bidirectional LSTM model captures the context and semantics that are then used by the CRF model to predict the correct labels. However, even though using neural networks instead of traditional machine learning does not require feature engineering, the time-consuming task of data labelling remains and might be more severe due to the amount of data that is required to train neural networks.

3.1.4 Suitability of Extraction Techniques

All three extraction techniques can be used successfully for attribute extraction. However, the suitability of the techniques is application dependent. Two dimensions that help determine the suitability are the text type and attribute type. In the following two paragraphs the dimensions are explained followed by details about how the dimensions help determine the suitability of extraction techniques.

The text type can be structured, semi-structured or unstructured [31][21]. Structured text has a fixed, pre-defined format. Examples are tables with product specifications such as the price, weight and colour on an e-commerce site. Semi-structured text has a flexible structure without a fixed format and contains natural language content. The content is commonly short and ungrammatical, but can also contain grammatical natural language. An example of semi-structured text is a HTML document of a web page containing HTML tags, a short and ungrammatical title in telegraph style and paragraphs of grammatical natural language. Unstructured text or free text, such as news articles, consists of natural language without a fixed format.

There are open and closed attributes, i.e. the set of values the attribute can take is open or closed. For open attributes, the set of possible values cannot be determined beforehand as there continue to remain unknown values or new values will continue to arise [8]. Examples are product names and prices. Cohen et al. [32] further

distinguished open attributes into 'regular sets' such as phone numbers, 'complex patterns' such as postal addresses and 'ambiguous patterns' such as person names. Closed attributes are characterized by the fact that the set of possible values is known or can be specified before the extraction model is developed. Example attributes are countries and currencies.

Figure 3.1 shows how the dimensions of text and attribute type help determine the suitability of extraction techniques.

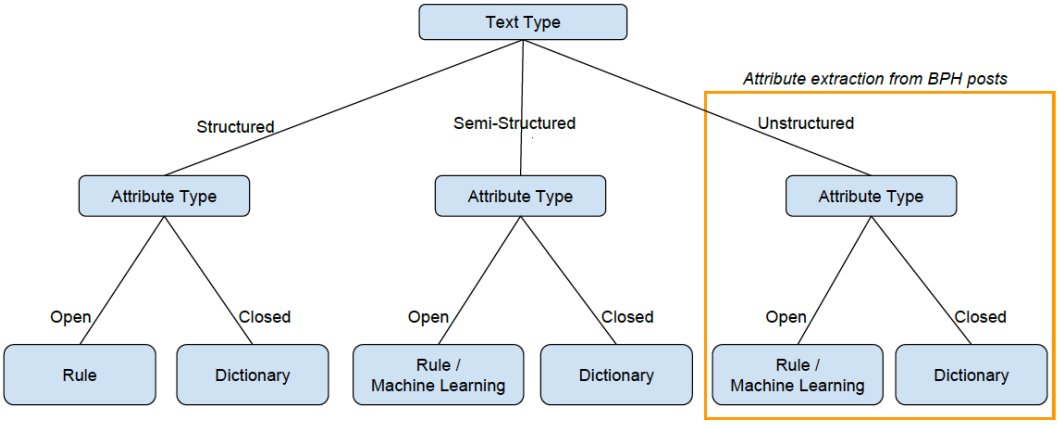


FIGURE 3.1: Applicability of extraction methods depending on text and attribute type

For all text types holds that the application of dictionary-based extraction will only result in a good performance for closed attributes, such as countries. In that case, a dictionary-based extraction can be sufficient and there is no need to spend time and resources to craft rules and patterns or to train a machine learning model based on labelled data.

For open attributes, such as product names, dictionary-based extraction does not provide a sufficient performance due to, e.g., a low recall caused by unknown values that were not recognized. Instead, a rule- and pattern-based extraction method is best applied in case of a structured text type. The reason is that the known structure of the text provides good indicators of attributes. For example, extracting price attributes from e-commerce tables when the price attribute has a fixed location in each table.

For open attributes in combination with a semi-structured or unstructured text type, either a rule- and pattern-based or a machine learning approach should be applied. This depends on the structure within the attribute and in the text surrounding the attribute. Regular set attributes, such as email addresses, that have a distinctive structure within the attribute can be extracted using a rule- and pattern-based approach. The same holds for complex pattern attributes that have a text context with a distinctive structure. An example is the price attribute that is commonly accompanied by a currency. For ambiguous attributes, such as product names, that have no distinctive structure, a manageable set of rules and patterns will not be able to capture the variability of the attribute values and their context. Instead a machine learning approach should be used.

3.2 Transfer Learning for Attribute Extraction

In this section, the field of transfer learning is explored with a specific focus on sequence labelling tasks, such as attribute extraction. First the field of transfer learning is introduced followed by an exploration of transfer learning for traditional machine learning and neural network based sequence labelling.

When applying machine learning, it is common to test and use a model on the same task and domain as it was trained on. This results in a model that performs well on, but is restricted to, a specific task and domain. In case a new model is required for a scenario where either the task or domain differs, a new model is trained from scratch requiring a sufficiently large labelled dataset to capture the new scenario. The goal of transfer learning is to overcome this dependency on task and domain by transferring knowledge learned by a pre-trained source model to the target model. Not only would this require less labelled target data it can also result in performance improvement and it requires less computational resources and time to develop new models [33]. Transfer learning is especially beneficial in case insufficient labelled data is available to train a target model from scratch [34][33].

Whereas the field of computer vision has received most attention from research into transfer learning, transfer learning has also effectively been applied to language processing tasks, such as attribute extraction. It can be applied to attribute extraction with traditional machine learning models and neural networks as described in the following two subsections.

3.2.1 Traditional Machine Learning

In contrast to neural networks, traditional machine learning models require extensive feature engineering. A type of transfer learning that has been applied to sequence labelling tasks with traditional models is using the results of pre-trained models as additional features. Using results such as cluster ID [10] as feature, provides knowledge learned by the source model. Examples of results that have been used as additional features are: cluster ID, topic ID based on latent Dirichlet allocation [35] and word representations such as features based on pre-trained word embeddings [36].

3.2.2 Neural Networks

Deep learning techniques have been effectively applied at sequence labelling tasks as described in section 3.1.3. Not only does it enable new opportunities for sequence labelling tasks, but also for effective transfer learning due to the hierarchical representation of the data through the layered set-up of deep neural networks [37]. Due to the difference in architecture, transfer learning for neural networks requires a different approach compared to traditional techniques. The most common approach is known as fine-tuning. That is also the approach used for this study. The fine-tuning approach makes use of a pre-trained source model, a neural network. The weights of the target model are then initialized according to the learned weights of the pre-trained model and fine-tuned during the actual training phase of the target model to fit the target task and domain [37][33]. When using the pre-trained weights instead of random weight initialization, the model has a head start in training that enables achieving a higher performance. Furthermore, it requires less labelled training data

compared to training from scratch.

The performance impact of transfer learning on neural network NLP applications in general depends on the source dataset size, the out-of-vocabulary words and semantic similarity with respect to the target dataset [37]. The out-of-vocabulary words are words that are in the target dataset and not in the source dataset. In practice, the suggested approach is to select a source dataset with a large vocabulary size and a low out-of-vocabulary metric instead of selecting a small dataset with a high semantic similarity [37].

Fine-tuning for NLP applications is commonly applied to the embedding layer of the neural network [33]. Alternatively, more layers can be transferred. However, this is a difficult task due to the risk of overfitting when using a small dataset and catastrophic forgetting of the learned knowledge when fine-tuning the model. Only recently, in 2018, Howard et al. [33] proposed a method named 'Universal Language Model Fine-Tuning' that avoids these issues for text classification. It was argued that this method can also be applicable for sequence labelling tasks.

3.3 Attribute Extraction and Transfer Learning on DNMs

Attribute extraction and transfer learning have been studied for a variety of domains. Whilst the successful techniques are domain independent, the results are not comparable due to differences in the domain and the attributes in them. Therefore, this section describes how the above mentioned techniques have been used in the two related studies about DNMs.

3.3.1 DNM Attribute Extraction

In a study performed by Portnoff et al. [9], darknet fora in three languages were used to train and evaluate new natural language tools to extract product and price information from DNM posts. This resulted in an accuracy of 0.866 for product extraction and a best F1-score of 0.988 for price information extraction. For product extraction, a SVM was trained to predict which noun phrase most likely represents the product advertised in the post. For this purpose, both surface and syntactic features of all tokens in the noun phrase were used in addition to the position of the noun phrase in the post. The SVM greatly outperformed a frequency baseline, selecting the most frequent noun or verb as product, and a dictionary baseline. The dictionary baseline used a dictionary of products obtained from the labelled training data. However, for individual fora, the product extractor was not able to achieve a high recall. The maximum recall reported was 0.70. For price extraction, two methods were developed and evaluated to extract the price, payment method and currency attributes from DNM posts. The first extraction method used regular expressions to extract all numbers and currencies. The second method was based on training a SVM to extract the price information using token count, position, POS and clusterID as features. The SVM greatly outperformed the regular expression based extractor. Using regular expressions resulted in a bad performance due to variety in the way price information is mentioned in posts and contextual information was not taken into account. This caused the regular expression to also match regular numbers. Furthermore, the majority of errors made by the SVM model were caused by ambiguity of words. For example, 'PM' can be meant as abbreviation for 'Personal Message' or

as payment method 'Perfect Money'. However, despite the high performance scores when training and testing on the same fora, precision and especially recall degrades largely when product and price extractors are tested cross-domain on a new forum. For example, the recall of the SVM price extractor dropped from 1.00 to 0.65 and precision from 0.98 to 0.84.

3.3.2 Transfer Learning for DNM Attribute Extraction

In a similar research performed by Durrett et al. [10], an initial attempt was made at performing transfer learning for DNM attribute extraction with traditional machine learning models. Without transfer learning, the cross-domain application of a product extraction SVM model on an unseen forum, resulted in a large performance reduction. The F1-score and Recall of 75.8 and 78.6 decreased to 50.6 and 50.2 respectively. Next, several attempts of fine-grained domain adaptation were performed to increase transferability of product extraction between darknet fora. The first attempt of transfer learning was based on adding cluster information, the brown cluster ID, to the feature vector of each token. This was aimed to enable the recognition of unknown products in the new domain. This did, however, not result in significant improvements. The second attempt was dictionary-based and added a feature indicating whether the token occurred in a gazetteer of known products. This resulted in slight improvements, though most were not significant. The last attempt had the largest impact and used a small amount of labelled target data during training. With 80 labelled target posts, the F1-score improved from approximately 56 to 71 using Hacker Forums as source and Darkode as target. However, it was concluded that more research should be conduction to explore the applicability of transfer learning for attribute extraction from DNM posts.

3.4 Summary

This section summarizes the related work and describes how it relates to our study.

Attribute extraction is the task of automatically discovering and extracting attributes from text. Three main extraction techniques can be distinguished: (i) dictionary-based extraction that uses a list of known attribute values to scan the text for matches, (ii) rule- and pattern-based extraction that uses linguistic extraction rules or patterns to locate attributes in text and (iii) machine learning based extraction that trains a model to automatically extract attributes using statistical techniques and a labelled dataset. For machine learning based extraction, traditional machine learning algorithms can be used as well as neural networks, such as the CRF and biLSTM-CRF respectively. Two dimensions that help determine the suitability of the extraction techniques for a particular application are the text and attribute type.

For our study, attributes such as the type of service, technical specification and contact details of the BPH provider were extracted from BPH posts. For this purpose, all three extraction techniques were applied and compared. To compare the machine learning models, we used both traditional machine learning models (MEM and CRF) as well as a neural network (biLSTM-CRF). BPH posts on DNMs have an unstructured text type and contain open and closed attributes. Therefore, for each attribute

the most suitable extraction technique was determined according to the considerations described in section 3.1.

When performing transfer learning, knowledge learned by a pre-trained source model is transferred to a new target model. One of the benefits of transfer learning is that it requires less labelled target data compared to training the target model from scratch. Transfer learning can be applied to traditional machine learning models by adding the output of the pre-trained model to the input data that is used to train the target model. Transfer learning can also be applied to neural networks. This approach is known as fine-tuning, where the weights of the pre-trained neural network are used to initialize the weights of the target model.

For our study, transfer learning was applied in an attempt to improve the CRF and biLSTM-CRF extractors. For this purpose, the output of pre-trained models was used as input data to train the CRF. For the biLSTM-CRF extractor, fine-tuning of the word and character embedding layers was performed and the impact several source datasets was compared.

Portnoff et al. [9] and Durrett et al. [10], performed initial studies on attribute extraction from DNMs and the applicability of transfer learning respectively. For attribute extraction, their scope was to extract product and price information from DNM posts of various topics. For this they used the three types of attribute extraction. The scope of our study is to extract all relevant attributes from DNM posts of one topic, BPH. For this we also used the three extraction techniques, but took it a step further by combining the techniques and comparing various types of machine learning algorithms including sequence models and neural networks.
Durrett et al. performed a type of transfer learning known as domain adaptation to reuse their SVM product extractor for another DNM. The focus of our project deviates from this as it aims to improve the performance of the developed machine learning based extractors. For this we also used transfer learning for traditional machine learning models, but took it a step further by applying transfer learning for neural networks.

Chapter 4

Theoretical Background

This chapter provides the theoretical background for this study. Section 4.1 describes the supervised classification techniques and algorithms that are relevant for this study. Section 4.2 describes transfer learning in more detail and section 4.3 describes the evaluation of sequence labelling tasks. Finally, section 4.4 provides a summary of the theoretical background.

4.1 Supervised Classification with Machine Learning

As defined by the artificial intelligence pioneer Arthur Samuel in 1959, machine learning is a "Field of study that gives computers the ability to learn without being explicitly programmed". Supervised classification is a class in the machine learning taxonomy that we will focus on for this research. In section 4.1.1, an overview of supervised classification is provided followed by a description of sequence labelling in section 4.1.2. Afterwards, sections 4.1.3 to 4.1.5 explain the MEM, CRF and LSTM algorithms including the biLSTM-CRF.

4.1.1 Supervised Classification

A classifier takes an input x and assigns it an output class y from a discrete set of possible classes Y. An example is a classifier that is trained to classify credit card transactions as genuine or fraudulent. Mapping the input x to an output y is based on a function that is learned during the training phase. In the training phase, a machine learning algorithm is used to iteratively optimize parameters of the mapping function based on example input-output (x,y) pairs, i.e. the training data. In each iteration, the weights are updated to minimize the errors made by the model. This is referred to as learning. The goal is to learn a function that correctly classifies unseen data when it is executed to perform the task is was designed for.

Focussing on probabilistic models, the classification model aims to, given an input, select the most probable class y^*, i.e. the class with the highest posterior probability. This is shown mathematically in equation 4.1:

$$y^* = \arg\max_y P(y|x) \tag{4.1}$$

The classification model can be generative or discriminative. For discriminative models, the function is optimized to model the conditional probability distribution $P(y|x)$ that is directly used to select the class y^*. For generative models, the function is optimized to model the joint probability distribution $P(x, y)$. In order to select class y^*, the joint probabilities are normalized to obtain $P(y|x)$. This is obtained by factorizing $P(x, y) = P(x|y)P(y)$ and using Bayes' rule shown in equation 4.2:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}, \text{where } P(x) = \sum_{y} P(x|y)P(y) \tag{4.2}$$

For the supervised classification methods used in this study, the input x is represented as features f_i, for $i \in \{1, 2, ..., n\}$, in a feature vector, $x = [f_1, ..., f_n]$. These features describe specific properties of the input x. Features can be nominal, such as the suffix of the token, boolean, such as 'ends with -ly', and numeric, such as the index of the token within a sentence. If the features are informative about the classification, the features help find the correct class y. For example, for POS tagging, knowing that the token ends with -ly is an indicator that it is more likely to be an adverb(Adv) than a noun(N) or adjective(Adj). For multi-class classification, these features are a function of the input x as well as the class y, known as feature function $f_i(x, y)$. As an example, the feature functions of the 'ends with -ly' feature for three classes are as follows:

$$f_i(x, Adv) = \begin{cases} 1, & \text{if } 'ends\ with\ \text{-}ly' = True \text{ and } y = Adv \\ 0, & \text{otherwise} \end{cases}$$

$$f_i(x, Adj) = \begin{cases} 1, & \text{if } 'ends\ with\ \text{-}ly' = True \text{ and } y = Adj \\ 0, & \text{otherwise} \end{cases}$$

$$f_i(x, N) = \begin{cases} 1, & \text{if } 'ends\ with\ \text{-}ly' = True \text{ and } y = N \\ 0, & \text{otherwise} \end{cases}$$

During training, a weight is assigned to each feature function to indicate its importance for assigning the correct class. Therefore, if feature 'ends with -ly' is very indicative of class 'adverb', the feature function gets a high weight.

4.1.2 Sequence Labelling

The supervised classification description above focussed on selecting an output y for each input x individually, referred to as regular classifiers. However, natural language tasks commonly use data sequences as input, e.g. a sequence of words that together form a sentence. Given a sequence of words denoted as $x = (x_1, x_2, ..., x_n)$, sequence labelling is the task of assigning a class y_i to each word x_i in the sequence x whilst assuming that the class of a word depends on the surrounding words and their classes [18]. This is a sequence of classification tasks, since each x_i is assigned an output class y_i from a discrete set of possible classes Y.

For sequence labelling, any classifier algorithm can be used. However, to take into account the dependencies of instances within a sequence, both neighbouring instances and predicted classes of these instances should be taken into account. Regular classifiers, such as SVM and MEM, can take neighbouring instances into account by including their features, referred to as context features, in the feature vector representation of the current instance. Sequence models, such as the CRF and HMM, also take the predicted labels of the neighbouring instances into account when predicting the class with the highest posterior probability. This makes sequence models highly suitable for sequence labelling tasks.

4.1.3 Maximum Entropy Model

The Maximum Entropy Model (MEM) is a regular supervised classifier that makes use of the information theoretic measure Entropy. Entropy is a measure for the uncertainty of a probability distribution. The intuition behind maximum entropy models is to model what is known and to not make any assumptions about the unknown. This is done by modelling the constraints learned from the training data, but otherwise keep the probability distribution as uniform as possible and therefore maximize the entropy. This first requires us to find the constraints and to find models that abide these constraints. Next, we have to find, from the set of abiding models, the most uniform model. That is the model with the maximum conditional entropy.

The constraints we want to model consist of the empirical expected values of the n binary feature functions, $f_i(x,y)$, using the empirical probability distribution $\widetilde{p}(x,y)$ learned from the training data. This is shown mathematically in equation 4.3. In this equation, N is the number of (x,y) pairs in the training data.

$$\mathbb{E}_{\widetilde{p}}[f_i(x,y)] \equiv \sum_{x,y} f_i(x,y)\widetilde{p}(x,y), \text{ where } \widetilde{p}(x,y) = \frac{1}{N} \times count(x,y) \tag{4.3}$$

Then, for each model p in the set of possible models P, holds that it abides the constraints if the model's expected value of the feature functions is equal to the empirical expected value. The subset C of models that abide the n constraints is defined as follows:

$$C \equiv \{p \in P \mid \mathbb{E}_p[f_i(x,y)] = \mathbb{E}_{\widetilde{p}}[f_i(x,y)] \text{ for } i \in \{1,2,...,n\}\} \tag{4.4}$$

To find the model p_* out of C with most uniform distribution, select the model that maximizes the conditional entropy H(p):

$$p_* = \arg\max_{p \in C} H(p) \tag{4.5}$$

With the conditional entropy H(P) defined as:

$$H(p) = H(Y|X) \equiv -\sum_{x,y} \widetilde{p}(x)p(y|x)\log p(y|x) \tag{4.6}$$

More information on the MEM algorithm can be found in the paper by Berger et al. [38] explaining the maximum entropy approach for NLP.

4.1.4 Conditional Random Field

The Conditional Random Field (CRF) model is a discriminative sequence model proposed by Lafferty et al. [39]. The CRF was developed to overcome limitations of the generative HMM and the discriminative MEMM, two popular sequence models. The following paragraph describes how the CRF avoids the limitations of the HMM and MEMM sequence models. Afterwards, the CRF model is described. Finally, the Linear Chain CRF that was used for this study is explained.

As described before, generative models model the joint probability distribution $P(x,y)$ over the feature vector represented input x and class y. This includes modelling p(x). However, this has several limitations[40]. These limitations are the potentially large dimensionality of x and the complex dependencies between the features of x. To

acknowledge these dependencies requires a large modelling effort and can lead to intractable models. However, assuming feature independence hurts the model performance. Discriminative models, such as MEMM and CRF, avoid modelling p(x) by directly modelling the conditional probability $P(y|x)$ over class y given input x for classification. Resulting in less complex models due to the fact that the complex dependencies between features of x do not need to be modelled, since they are a constant with respect to class y. The MEMM does, however, have a label bias problem due to local normalization of probabilities. This causes a bias towards states with fewer transitions to other states [39]. This problem is avoided by the CRF due to global instead of local normalization of probabilities.

The CRF is an undirected discriminative graphical model. The conditional probability p(y | x) is modelled using a graph consisting of nodes for Y and X with undirected edges between them. Y is the class sequence and X is the sequence of observed input vectors. The definition of the model given by Lafferty et al.[39] is as follows:

> Let $G = (V, E)$ be a graph such that
> $\boldsymbol{Y} = (\boldsymbol{Y}_v)_{v \in V}$, so that $\boldsymbol{Y}$ is indexed by the vertices of G. Then $(\boldsymbol{X}, \boldsymbol{Y})$ is a conditional random field when the random variables $\boldsymbol{Y}_v$, conditioned on $\boldsymbol{X}$, obey the Markov property with respect to the graph: $p(\boldsymbol{Y}_v|\boldsymbol{X}, \boldsymbol{Y}_w, w \neq v) = p(\boldsymbol{Y}_v|\boldsymbol{X}, \boldsymbol{Y}_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.

The Linear Chain CRF is a family of CRF models that has been successfully used for sequence labelling tasks. The graphical representation is shown in figure 4.1.

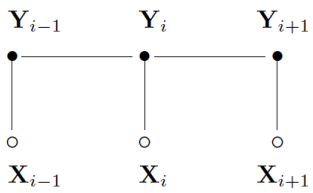


FIGURE 4.1: Graphical representation of the Linear Chain CRF [39]

The conditional probability $p(\mathbf{y}|\mathbf{x})$ is calculated as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}, \tag{4.7}$$

where $\{f_k(y, y', x_t)\}_{k=1}^{K}$ is a set of feature functions, θ_k is the feature function weight, and $Z(x)$ is the normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} exp \left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\}. \tag{4.8}$$

The feature function $f_k(y_t, y_{t-1}, x_t)$ ranges over all possible transition and state feature functions. A transition feature function captures a specific transition from a class at timestep t to a class at t-1 ($f_{y_t y_{t-1}}$). A state feature function captures the occurrence of a specific combination of observation and class at the same timestep t ($f_{o_t y_t}$). The observations refer to the features in the feature vector $\mathbf{x}_t$, a vector containing the features from the current observation and, when applied, the context features of neighbouring observations. An example $f_{o_t y_t}$ and $f_{y_t y_{t-1}}$ are shown below. The $f_{o_t y_t}$

feature function example is equal to one of the feature function examples for supervised classification in section 4.1.1.

$$f_{o_t y_t} = \begin{cases} 1, & \text{if } 'ends\ with\ \text{-}ly' = True \text{ and } y_t = Adv \\ 0, & \text{otherwise} \end{cases}$$

$$f_{y_t y_{t-1}} = \begin{cases} 1, & \text{if } y_t = Adj \text{ and } y_{t-1} = Noun \\ 0, & \text{otherwise} \end{cases}$$

4.1.5 Long Short-Term Memory Neural Network

For many sequence labelling applications including named entity recognition, the state-of-the-art performance is achieved using Long Short-Term Memory (LSTM) based models [41]. The LSTM is a Recurrent Neural Network (RNN) proposed by Hochreiter et al. [42]. This section first shortly explains neural networks and recurrent neural networks. Afterwards, we explain the LSTM, how it differs from regular RNNs and why these changes make the LSTM especially suitable for sequence labelling.

An artificial neural network is a weighted directed graph consisting of an input layer, output layer and one or more hidden layers. Each layer consists of artificial neurons, i.e. the nodes, with weighted directed edges between them allowing the data to flow through the layers. A neuron receives data input from incoming edges, performs a simple function and sends the output along the outgoing edge. Combining these neurons with this architecture enables the artificial neural network to perform complex tasks. Neural networks can be distinguished in two types: feedforward and recurrent neural networks [43]. For feedforward neural networks hold that the output of a neuron in a layer always moves forward to the next layer and therefore does not affect the current layer. For RNNs this does not hold, allowing outputs to loop back to the same layer such that information can persist in the network.

RNNs make use of their internal state, often referred to as memory, to store information about calculations from previous inputs. Therefore, this information is accessible when performing calculations on subsequent inputs. This characteristic makes them suitable for sequence labelling. As depicted in figure 4.2, a neuron receives x_t, the input a time step t, and s_{t-1}, the hidden state or memory from preceding time steps, to calculate o_t, the output at timestep t. At each time step, the hidden state is updated with a function of x_t, s_{t-1} and the weights of the directed edges such that $s_t = f(\mathbb{U}x_t + \mathbb{W}s_{t-1})$.

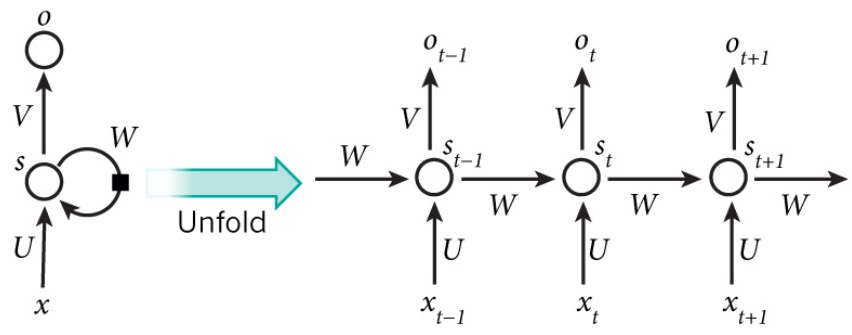


FIGURE 4.2: RNN neuron[1]

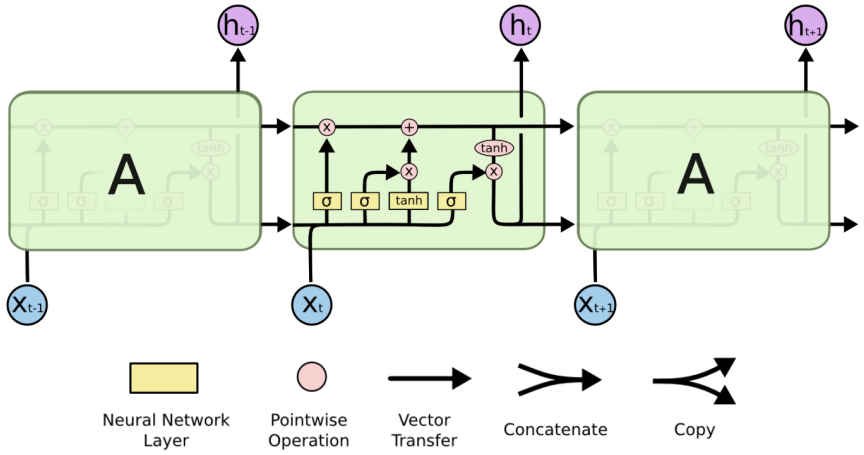
In theory, RNNs can apply stored information from any time step in the past, i.e. a long-term memory and a short-term memory. However, in practice this is not the case due to the vanishing and exploding gradient problem during the training phase. In the training phase, the weights of the RNN are iteratively updated to minimize errors made by the model using backpropagation through time. Assume a sequence consisting of tokens, i.e. the time steps. Each iteration starts with forward propagation. This results in the total error for the sequence by summing over the error of each time step. Next, with backpropagation through time, the calculated error is back-propagated trough the unrolled RNN to update the weights. This is done using the error gradient, i.e. the partial derivative of the error with respect to the weights, at each state s_t. This includes multiplications of gradients due to the chain rule. For example, the gradient at t=1 is multiplied by the gradient of all later time steps. This is where the problem of the vanishing gradient comes is, because multiplying values below 1.0 results in a gradients becoming closer and closer to 0, the more timesteps are included. This means that for long sequences, i.e. many multiplications, the weights in the earlier layers of the network are not updated, i.e. no learning takes place. In contrast, the values explode and the weights are updated too much if the gradients are larger than 1.0.

Long Short-Term Memory networks, first proposed by Hochreiter and Schmidhuber [42], resolves the gradient problems and enables learning long-term dependencies. Similar to regular RNNs, LSTM networks can be unrolled into a chain of gated cells for each time step in the sequence as shown in figure 4.3a. The difference is the way information is selectively removed, added and outputted from memory. This is represented by the upper line, called the cell state, going through each cell. These changes to the cell state are based on the relevance of information that the forget gate, input gate and the output gate learned to determine in the training phase. We will now explain each step in the gated cell at timestep t in more detail.

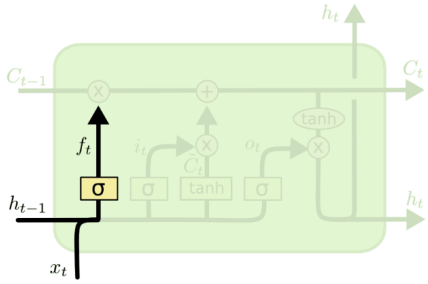
Starting with the forget gate, shown in figure 4.3b. The input x_t and output from previous gated cell h_t are concatenated, represented as $[x_t, h_t]$, and used as input vector for the first sigmoid layer. The sigmoid layer transforms the values of the input vector to values between 0 and 1. With pointwise multiplication, shown in figure 4.3d, between the obtained vector f_t and the previous cell state vector, C_{t-1}, cell state values are multiplied by values between 0 and 1, were 0 means the value is completely forgotten and 1 means the value remains unchanged.

Next, we determine what information from $[x_t, h_t]$ to add to the cell state as depicted in figure 4.3c. The input gate, a sigmoid layer, converts $[x_t, h_t]$ to values between 0 and 1, representing the importance of each value. The tanh layer converts $[x_t, h_t]$ to values between -1 and 1. This regulates the network by avoiding exploding (and extremely small) values. The pointwise multiplication, shown in figure 4.3d, of the two resulting vectors ensures that less relevant values are multiplied by a value closer to 0. This prevents irrelevant information from being added to the memory. Now that we know what information to add to the cell state, the vector is added to the cell state using pointwise addition.

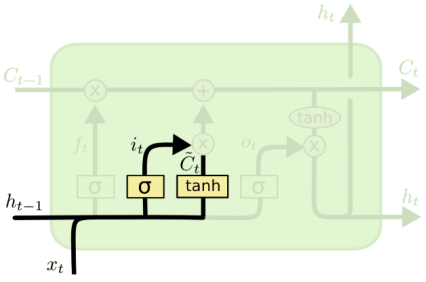
[1]Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/



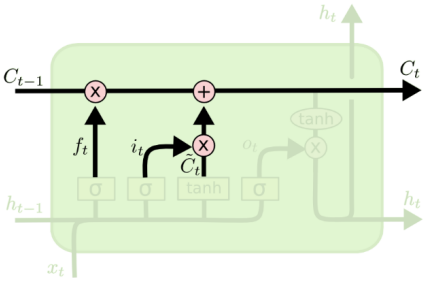
(A) Unrolled LSTM network



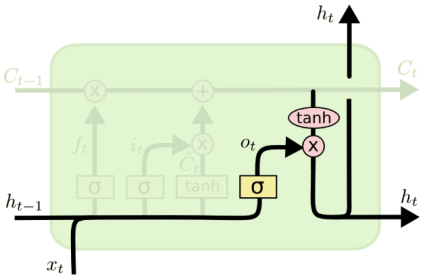
(B) Forget gate



(C) Input gate



(D) Update cell state



(E) Output gate

FIGURE 4.3: LSTM network with gated cells[2]

Lastly, the output is generated based on the updated cell state as shown in figure 4.3e. To determine which values of the cell state to output, the $[x_t, h_t]$ goes through the sigmoid function of the output gate for conversion to values between 0 and 1. At the same time, the cell state values are transformed to values between -1 and 1 in the tanh layer. Both vector are pointwise multiplied to find the cell state values that we want to output h_t. h_t is the output for timestep t and is used as input for the next gated cell together with the actual cell state.

Bidirectional LSTM-CRF

The bidirectional LSTM-CRF (biLSTM-CRF), proposed by Huang et al. [44], combines a bidirectional LSTM network with the CRF algorithm. In the following paragraph we explain how the LSTM network collaborates with the CRF and the benefit

[2]Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

of using a bidirectional LSTM.

As described in section 4.1.4, CRFs make use of feature functions to predict the most likely class labels for an input sequence. When using an LSTM-CRF, the most likely class label is predicated based on the LSTM output instead of feature functions. The LSTM therefore replaces the feature engineering step that is otherwise required for the stand-alone CRF. The LSTM provides an output for each token x_i in sequence x where i = [1,...,n]. These outputs are based on the current token and on relevant information from preceding tokens, but not on following tokens. This is where the bidirectional LSTM is helpful. For this network, two LSTM networks are trained. The first model receives the original sequence $x = [x_1, ..., x_n]$ as input and the second model receives the reversed sequence $x = [x_n, ..., x_1]$. Then, for each token, the output of both layers is combined. This combines relevant information from preceding as well as following tokens. The combined output is then used as input of the CRF. In this collaboration, the LSTM ensures relevant characteristics from context tokens are included in predicting the correct labels and the CRF includes information about previously predicted labels.

4.2 Transfer Learning

In this section the definition of transfer learning is provided. Furthermore, the three main categories of transfer learning are described.

4.2.1 Notations and Definitions

For transfer learning, the following notations and definitions apply:
Source: $\mathcal{S}$ with source task $\mathcal{T}_\mathcal{S}$ and source domain $\mathcal{D}_\mathcal{S}$
Target: $\mathcal{T}$ with target task $\mathcal{T}_\mathcal{T}$ and target domain $\mathcal{D}_\mathcal{T}$
Task: $\mathcal{T} : \{\mathcal{Y}, f(\cdot)\}$ with $\mathcal{Y}$ representing the label space, and the predictive function $f(\cdot)$ of assigning labels $\mathcal{Y}$, which is to be learned from the data.
Domain: $\mathcal{D} : \{\mathcal{X}, P(X)\}$ where $\mathcal{X}$ is the input feature space, and $P(X)$ the marginal probability distribution of the input.

Using the notation given above, transfer learning is defined as follows:
"Given a source domain $\mathcal{D}_\mathcal{S}$ and learning task $\mathcal{T}_\mathcal{S}$, a target domain $\mathcal{D}_\mathcal{T}$ and learning task $\mathcal{T}_\mathcal{T}$, transfer learning aims to help improve the learning of the target predictive function $f_\mathcal{T}(\cdot)$ in $\mathcal{D}_\mathcal{T}$ using the knowledge in $\mathcal{D}_\mathcal{S}$ and $\mathcal{T}_\mathcal{S}$, where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$, or $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$." [45]

4.2.2 Transfer Learning Categories

Various categories of transfer learning can be distinguished. In a survey on transfer learning performed by Pan et al. [45], transfer learning is divided in three main categories based on the different changes of target $\mathcal{T}$ with respect to source $\mathcal{S}$ as described above. The categories are inductive transfer learning, transductive transfer learning and unsupervised transfer learning as shown in table 4.1 together with their indicators.

Inductive Transfer Learning

For inductive transfer learning the tasks differ ($\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$) whereas the domains may or may not differ. Furthermore, labelled target data should be available. Depending

	Labelled		
Category	$\mathcal{D}_\mathcal{T}$	$\mathcal{D}_\mathcal{S}$	**Similar To**
Inductive $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$ and $\mathcal{D}_\mathcal{S} = \mathcal{D}_\mathcal{T}$ or $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$	Yes	Yes	Multi-task learning, if $\mathcal{T}_\mathcal{S}$ and $\mathcal{T}_\mathcal{T}$ are learnt simultaneously.
	Yes	No	Self-taught learning
Transductive $\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{T}$ and $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$	No	Yes	Domain adaptation, if $\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{T}$ and $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$ Covariate Shift, if $\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{T}$ and $\mathcal{D}_\mathcal{S} = \mathcal{D}_\mathcal{T}$
Unsupervised $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$	No	No	

TABLE 4.1: Transfer learning categories

on the amount of labelled source data, inductive transfer learning relates to two specific fields: multi-task learning and self-taught learning. With multi-task learning sufficient labelled source and target data are available and a model can be trained for multiple tasks simultaneously. With self-taught learning no labelled source data is available.

An example of inductive transfer learning is fine-tuning the embedding layer of a biLSTM-CRF model aimed to improve the model performance. This method transfers knowledge from an embedding model pre-trained on a large corpus to induce, e.g., an attribute extractor such as the one of our study.

Transductive Transfer Learning

For transductive transfer learning the task is equal ($\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{T}$) but shifted to a different domain ($\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$) for which no sufficient labelled target data is available. Transductive transfer learning relates to domain adaptation.

An example of domain adaptation is the transfer learning attempt of Durrett et al. [10]. Here they reused the attribute extraction model pre-trained on one DNM to perform attribute extraction on another DNM. Another example is cross-lingual transfer learning that makes use of correspondence between languages to transfer knowledge from one language to another [11].

Unsupervised Transfer Learning

For unsupervised transfer learning the tasks differ ($\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$) and there is no labelled source and target data available. This type of transfer learning can be used for unsupervised learning tasks, such as clustering and dimensionality reduction.

4.3 Evaluation

Evaluation of attribute extraction models can be a complex task. The main cause of the complexity is the fact that attributes can consists of more than one token and that the exact number of tokens, the boundaries of the attribute, is not fixed. The

complexity of deciding which tokens are part of attributes is portrayed in the fact that annotators also disagree on this decision.

Due to the fact that an attribute can consist of multiple tokens and each token has its own label, five different types of predicting errors can occur [25] as shown in figure 4.4. Based on these predicting errors, an extracted attribute can match the ground truth in one of three ways: an exact match, partial match or no match. Exact matches occur when no predicting error is made and both boundaries and label are correctly predicted. Partial matches occur with predicting error three and four where either the label or the boundaries are incorrect. No match occurs when both boundaries and labels are incorrect as in predicting error one, two and five.

	Error type	True labels	Predicted labels
1	Non-attribute recognized as attribute	Service: Dedicated and virtual servers in	Service: Dedicated and virtual servers; Service: in
2	Attribute recognized as non-attribute	Service: Dedicated and virtual servers in	Dedicated and virtual servers in
3	Correct boundaries and wrong label	Service: Dedicated and virtual servers in	Technical specification: Dedicated and virtual servers in
4	Wrong boundaries and correct label	Service: Dedicated and virtual servers in	Dedicated and Service: virtual servers in
5	Wrong boundaries and wrong label	Service: Dedicated and virtual servers in	Dedicated and Technical specification: virtual servers in

FIGURE 4.4: Types of predicting errors

Various evaluation models have been introduced in the past and each model has its own method of assigning scores based on the predicting errors. A simple but strict evaluation model was used for the CoNLL-2003 shared task [28] where only exact matches are seen as correct and all other matches are incorrect. Other evaluation models take the possibility of partial matches into consideration. The Message Understanding Conference (MUC) evaluation model [46] is a frequently used model that takes partial matches into account by scoring the extraction model separately for label and boundaries.

4.3.1 Performance Measures

According to the MUC evaluation, type (label) and text (boundary) are correct in the following scenarios: "A correct TYPE is credited if an entity is assigned the correct type, regardless of boundaries as long as there is an overlap. A correct TEXT is credited if entity boundaries are correct, regardless of the type." [25]. For both the evaluation of the type and text, three measures are calculated. These are the number of correct predictions (COR), the number of actual predictions made by the model (ACT) and number of possible correct predictions (POSS). Using these measures, the precision and recall can be calculated for the type and text combined. This enables the calculation of a final performance score, a micro-averaged F_β-score.

Precision (P) is the percentage of actual predictions made that are correct [46].

$$\mathrm{P} = \frac{\mathrm{COR}_{type+text}}{\mathrm{ACT}_{type+text}} \tag{4.9}$$

Recall (R) is the percentage of possible correct predictions that are correctly predicted [46].

$$\mathrm{R} = \frac{\mathrm{COR}_{type+text}}{\mathrm{POSS}_{type+text}} \tag{4.10}$$

F_β-score is the weighted harmonic mean of precision and recall. For the micro-averaged F_β-score, precision and recall are calculated using sums of the individual COR, ACT and POSS measures of type and text. Therefore, each extracted attribute has an equal weight for the evaluation. A macro-averaged F_β-score gives each class an equal weight by calculating the scores for all classes and taking the unweighted mean [47]. An alternative is the weighted macro-averaged F_β-score that calculates the scores for all classes and takes the weighted average based on class size.

$$\mathrm{F}_\beta\text{-score} = (1+\beta^2) \cdot \frac{\mathrm{P} \cdot \mathrm{R}}{(\beta^2 \cdot \mathrm{P}) + \mathrm{R}}$$

Here β represents the relative importance of recall over precision. The F1-score (β = 1) assigns equal importance to both precision and recall. For this application recall is considered of greater importance than precision. Therefore one of two possible alterations can be made. The first alteration is to report both precision and recall in addition to the F1-score. Another alteration is to increase β of the F-score [46].

4.4 Summary

Sequence labelling is a classification task that, given a sequence of instances such as words, assigns a class to each word in the sequence whilst assuming that the class of a word depends on the surrounding words and their classes. Supervised machine learning algorithms that can be used for this are regular classifiers, such as the Maximum Entropy Model, sequence models, such as the Conditional Random Field, or neural networks, such as the Long Short-Term Memory based Bidirectional LSTM-CRF. Transfer learning can be split in three main categories. These categories are inductive transfer learning, transductive transfer learning and unsupervised transfer learning. The category depends on the difference between target T and source S. A popular evaluation model for sequence labelling tasks, such as attribute extraction, is the MUC evaluation model. This model takes the possibility of partial matches into consideration.

Chapter 5

Dataset

This chapter introduces the dataset. Section 5.1 describes the dataset construction. Section 5.2 describes the results of an exploratory data analysis. Section 5.3 describes the data preparation steps. Section 5.4 describes the attribute identification and shows the results. Section 5.5 describes the manual annotation that was performed.

The data this study focussed on originates from a DNM. The dataset consists of a subset of 844 multilingual BPH posts from 27 threads that were manually identified to have BPH as subtopic. BPH posts refers to posts in threads with BPH as subtopic and that were written by the BPH provider. In section 2, two example BPH posts were presented in figures 2.2 and 2.3.

The raw data consists of the post text, author, thread ID, thread title, publication date in UTC, category, language and a machine translation. As is common for DNMs [16][17], the main language is Russian followed by English. BPH posts, from here on referred to as posts, can contain Bulletin Board Code (BB-code) to format the content, e.g. [b] and [size=5]. Furthermore, a post can consist of multiple languages, e.g. a Russian post with an English version and vice versa. Another characteristic of Russian posts is the usage of English key words, e.g. 'Hosting' and 'Pentium 4 3.0GHz', to make the post more intuitive for readers that are unable to read the original language or in case the term does not exist or is uncommon in the original language.

5.1 Dataset Construction

The starting point for the dataset construction consisted of two parts. The first part was a DNM user list. The list contains for each user the probability of being a BPH provider given the words used in their post. These probabilities were obtained in a previous project and used a gazetteer of pre-determined distinctive words for BPH and other DNM topics. The second part was the set of raw posts from the DNM.

The aim was to extract the BPH posts written by the thread initiator, the BPH provider. Therefore, the user list was sorted to show users most probable to have a connection with BPH first. Using the sorted user list, a domain expert labelled users as 'provider', 'non-provider' or 'unknown'. Here provider refers to BPH provider. For users labelled as 'unknown', the corresponding initiated threads were manually inspected. In case one or more BPH advertisements were created by the user, the user was labelled as 'provider' and 'non-provider' otherwise. The manual inspection continued until ten subsequent users were labelled as 'non-provider'.

The threads from users labelled as 'provider' were then manually inspected to detect and subsequently extract threads with BPH as subtopic. Since the focus is on advertisements of BPH services, the resulting BPH threads and their posts were further reduced to only include posts that were written by the BPH provider. Posts from other users containing, e.g., comments, reviews or questions were omitted.

5.2 Exploratory Data Analysis

This section describes the results of an exploratory data analysis aimed to gain insight into the constructed dataset of 844 posts written by 23 BPH providers. In this section, the distribution of individual variables and the relationships between variables are explored. The variables included are post length, post language, thread post count, thread category and publication day and time.

The post lengths vary between one and 1800 tokens, splitted on white space, with a mean of 28 and median of ten tokens. The post length distribution over all 844 BPH posts including mean and median is shown in the Tukey boxplot in figure 5.1. Noticeable are the two rightmost outliers with token counts 936 and 1787 and the fact that half of the posts contain ten tokens or fewer. After a manual inspection of posts with ten tokens or fewer, it turned out that the set of 298 posts with fewer than five tokens contained almost no attributes and should be excluded from that dataset. In contrast, posts with five up to ten tokens are relevant, since they do commonly contain attributes.

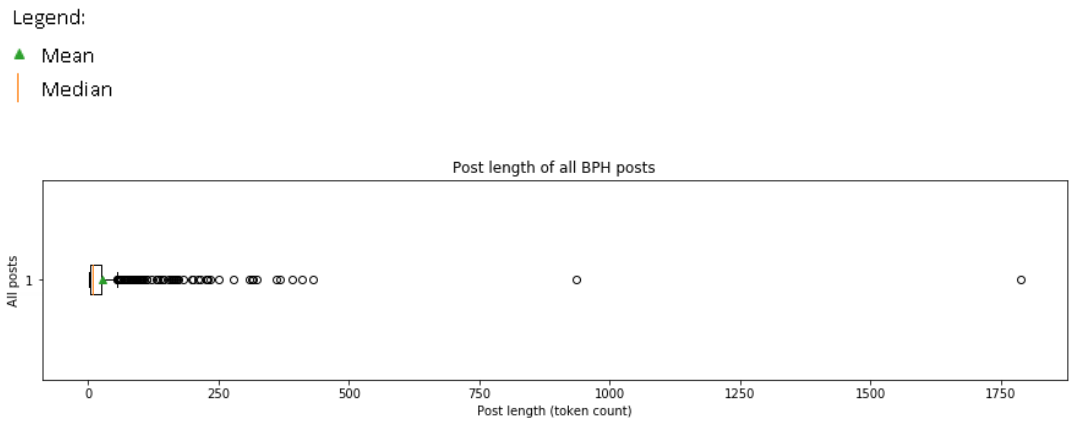


FIGURE 5.1: Post lengths of the 844 BPH posts [mean: 28, median: 10]

In the dataset, posts of fourteen different predicted languages are present. The two most common languages are Russian and English with 517 and nineteen posts of five or more tokens respectively. The remaining languages occurred in one up to fifteen posts. Figure 5.2 shows for each language the number of posts with five or more tokens and with fewer than five tokens. Noticeable is the fact that many sparse languages, such as Ukrainian(uk) and Albanian(sq), consists solely of posts with fewer than five tokens. Due to lack of text, the language could have been wrongly predicted, causing the languages to be included due to an error. Furthermore, the label '??' indicates that for some posts no language could be predicted. Important for this research is the fact that all languages besides Russian and English contain fewer than five posts with five or more tokens. Therefore, it was argued that only Russian and English posts should be used for this research, since no reliable extraction models

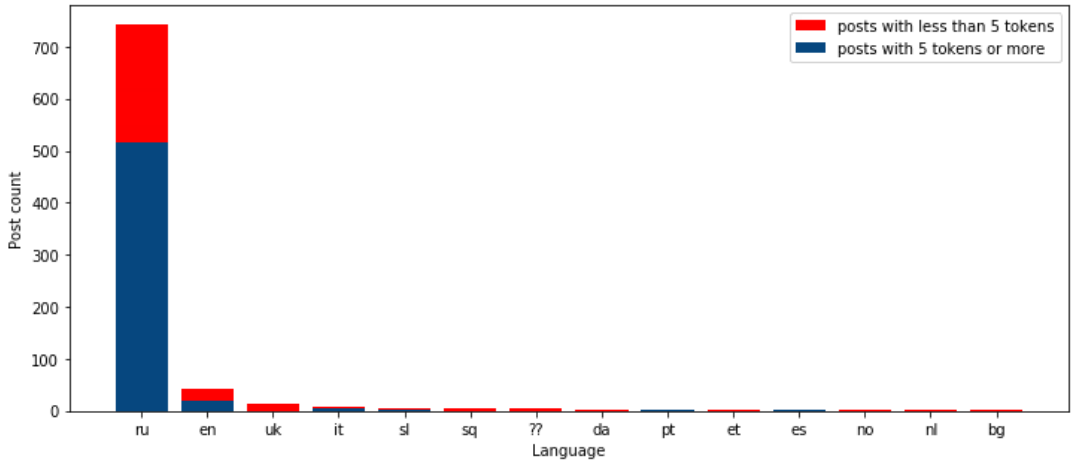


FIGURE 5.2: Post count per Language

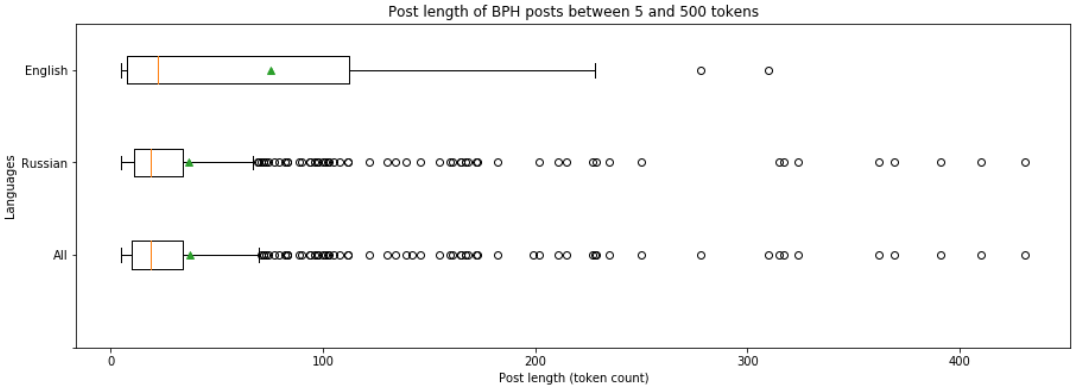


FIGURE 5.3: Length of English, Russian and all posts with five or more tokens

can be developed in case of too little data. Besides the varying languages assigned to posts, it was found that posts themselves can contain multiple languages. For example, Russian posts can contain an English translation provided by the author. Splitting posts in case they consist of two separate versions can therefore enable a more distinct split of Russian and English posts. Furthermore, in case many Russian posts contain an English translation, the number of English posts can grow substantially.

Based on the findings above, figure 5.3 shows the post length distribution for posts with five or more tokens for all languages and for Russian and English posts separately. Furthermore, the two rightmost outliers were removed to get a clearer picture of the distribution. However, they remain part of the dataset. Due to the high percentage of Russian posts in the complete dataset, the Russian post length distribution is similar to that of all languages combined. Comparing the distributions, English posts have a higher median and the average post is longer than Russian posts. Furthermore, English posts have fewer outliers.

The 27 threads vary in number of posts as shown in figure 5.4. The majority of threads have five (median) or fewer posts. There were, however, some active threads with a post count up to 268, causing a disproportionate representation of authors in the dataset. This can be argued to be an influencing factor, since posts of one author

can have distinct characteristics due to a particular writing style of the author.

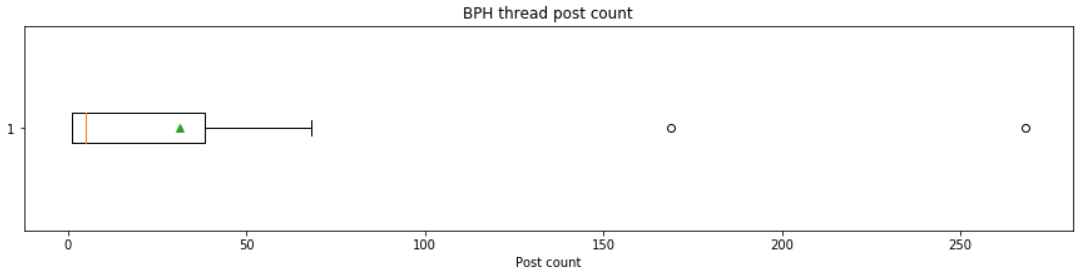


FIGURE 5.4: BPH thread post count

When creating a new thread on a DNM, the creator can assign it a category. Looking at the categories of the 27 threads, all but one thread are assigned the category that encompasses hosting and related services. This category is assigned to two percent of all posts on the DNM. The remaining thread in the constructed dataset has the 'general' category.

We explored post lengths for all languages and for English and Russian separately. Next, post lengths were explored per post index or position in the threads to, e.g., see differences between first and second posts in BPH threads. The two outmost outliers were excluded from the visualisation. The results are presented in figure 5.5. Not all threads have an equal number of posts. Therefore, there will be more posts with index 1 than with index 30. In figure 5.5, the post length distributions are shown for posts up to index 19, since for each remaining index there are fewer than 10 posts in the dataset. The blue line shows post length of five tokens. Noticeable is the difference in post length when comparing first posts with all following posts. Furthermore, all first posts have five tokens or more and all subsequent post indexes have a median of at least five.

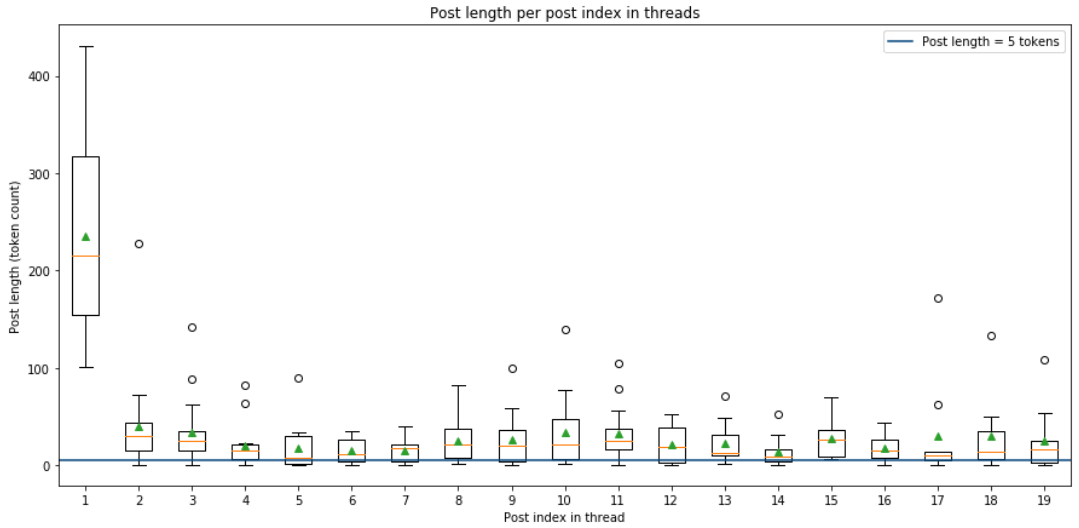


FIGURE 5.5: Post length per post index in BPH threads

Lastly, some insight is given into the publishing of BPH posts on the DNMs by showing the number of posts per day of week in figure 5.6 and per time of day in figure

5.7. Noticeable is the fact that, in general, the number of posts on a day reduces as the week progresses, with the least number of posts on Sunday. Furthermore, the most popular time is from 8AM to 12AM UTC.

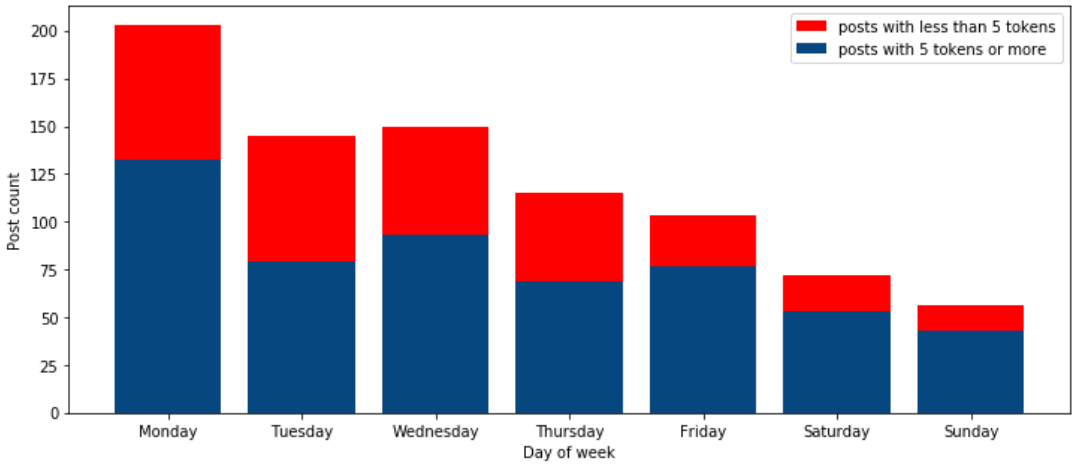


FIGURE 5.6: Post count per day of week

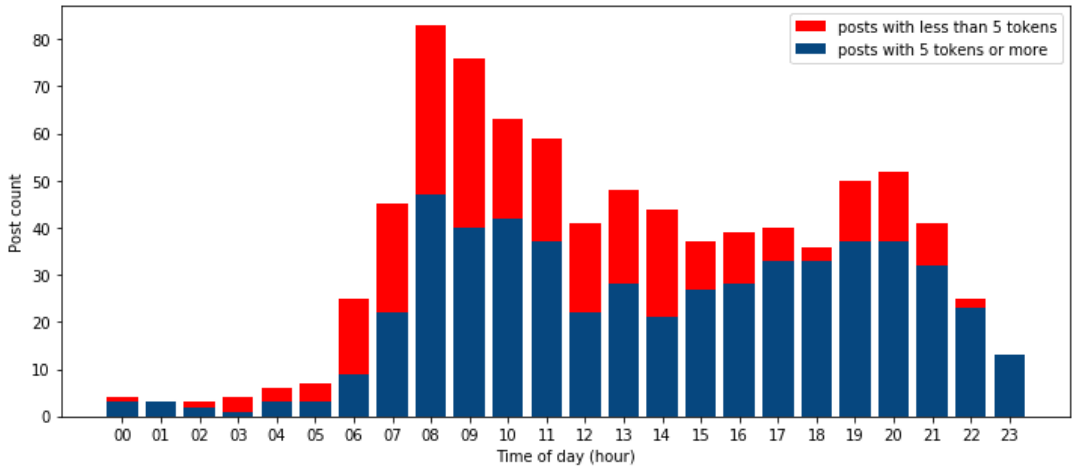


FIGURE 5.7: Post count per time of day (UTC)

5.3 Data Preparation

Using the constructed dataset and the findings from the exploratory data analysis, the dataset was prepared for our study. This included removal of the posts with scarce languages labels and posts consisting of fewer than five tokens. Furthermore, duplicate posts were removed and posts were split in an English and a Russian dataset. No data pre-processing steps, e.g. lowercasing and stemming, were performed at this stage.

The duplicate posts were removed using the hashlib Python library[1]. The duplicate files were detected by calculating and comparing the MD5 hash for each file.

[1]Hashlib library: https://docs.python.org/3/library/hashlib.html

5.3.1 Automatic Split of English and Russian Posts

To enable the development of separate English and Russian extraction models, a distinct Russian and English dataset is required. Therefore, the constructed dataset was split in two phases. In the first phase, posts were split in two groups based on the occurrence of either language label 'ru' or 'en'. The second phase is based on the finding that Russian posts can contain an English translation provided by the author and vice versa. Therefore, posts with both languages were automatically detected and split. The part containing the second language was then added as post to the other dataset to keep English and Russian separate. The exact method for the second phase is described below.

The second language can occur as a translation subsequent to the original text or it can be intertwined with the original language. In order to avoid disturbing the structure of the posts, only posts with a distinct Russian and English version were automatically split and the intertwined posts remained unaltered. The split was performed based on the language of the paragraphs. The paragraphs were obtained by splitting the post using the regular expression $\backslash n(\backslash s| - |\backslash r|\backslash n)^*\backslash n$. Using the PyPi langdetect library[2], the language of each paragraph was detected. This resulted in a sequence of languages representing a post. For example, a post consisting of two Russian paragraphs followed by two English paragraphs was represented by ['Ru', 'Ru', 'En', 'En'].

For each paragraph, the most likely associated languages were obtained. In case either Russian or English was detected, the paragraph was assigned the respective language. In case neither or both Russian and English was detected, the presence of Cyrillic script was used as indicator of Russian paragraphs. Otherwise, in case no Cyrillic script was found, the paragraph was labelled as English. Furthermore, paragraphs without alphabetic characters were omitted from the language sequence due to the inapplicability of language detection. An example is the paragraph representing a border between the Russian and English versions '———-'.

Using the language sequences, only posts with a distinct Russian and English version were automatically split. Therefore, a requirement for splitting a post was the presence of a language sequence that reflects this separation, such as ['Ru', 'Ru', 'En', 'En']. Posts that did not have approximately 50% Russian paragraphs followed by approximately 50% English paragraphs or vice versa remained unaltered.

5.3.2 Final Russian and English Dataset

The originally constructed dataset consisted of 517 Russian and nineteen English posts. After the duplicate post removal and splitting the data set as described above, the final Russian and English datasets consisted of 470 and 62 posts respectively. Table 5.1 shows the number of posts, paragraphs and sentences in both datasets. Sentences were obtained by splitting posts on newlines ($\backslash n$).

[2]PyPi langdetect library: https://pypi.org/project/langdetect/

	Russian	English
Posts	470	62
Paragraphs	1358	160
Sentences	2960	333

TABLE 5.1: Post, paragraph and sentence count in the Russian and English datasets

5.4 Attribute Identification

Using the final Russian and English datasets, the domain was explored to identify generic and BPH specific attributes. For this purpose, posts were manually inspected. Due to the manageable size of the English dataset, all posts were included for inspection. For Russian posts, a domain expert with knowledge of the language performed the attribute identification task on a set of approximately one hundred posts. Afterwards, the findings were combined and then verified by the domain expert. For each identified attribute, the most suitable extraction method was then determined based on attribute type as described in section 3.1.4 and taking into account the unstructured text type of BPH posts.

The attribute identification resulted in the discovery of sixteen attributes. In table 5.2, each attribute is presented including the data type and examples. Due to the sensitivity of information, some examples were anonymized or omitted. Furthermore, the generality of the attribute is included to distinguish between generic attributes and attributes specific for BPH posts. The attribute type (value set column) and the most suitable extraction method are shown in the last two columns.

Closed Attributes

Three of the identified attributes have a closed value set. These are payment period, currency and location. For both location and payment period, fixed dictionaries can be constructed that capture all locations worldwide and all time periods. However, due to the growing number of existing cryptocurrencies, the currency attribute is not as fixed. Possible solutions for currency extraction are to frequently update the dictionary with newly discovered cryptocurrencies or to use machine learning based extraction.

Open Attributes

The remaining attributes with an open value set were further divided in regular sets, complex patterns and ambiguous patterns. Attributes with a regular set are jabber mail, ICQ and URL due to their distinct internal structure. The price(amount) attribute in BPH posts is of the category complex patterns. Prices commonly consist of integers that do not have a distinct internal structure. However, they commonly occur near a currency or payment period. The four attributes mentioned in this paragraph have a distinct internal or external structure and that makes them eligible for rule- and pattern-based extraction.

Attributes with ambiguous patterns are: content, communication channel, nickname, other contact details, payment method, service, service details, support and

technical specification. Due to the lack of structure, machine learning based extraction is the most suitable extraction method.

Attribute	Data Type	Example	Generic	Value Set	Extraction Method
Content	String	exploits, botnets	N	Open	Machine Learning
Communication Channel	String	jabber, ICQ	Y	Open	Machine Learning
ICQ	Int/String	000000, 000-000-000	Y	Open	Rule and pattern
Jabber Mail	String	123-email@email.ru	Y	Open	Rule and pattern
Location	String	Europe, Азии (= Asia)	Y	Closed	Dictionary
Nickname	String	<nickname>	Y	Open	Machine Learning
Other Contact Details	Int/String	<contactdetails>	Y	Open	Machine Learning
Payment Method	String	moneybookers	Y	Open	Machine Learning
Price - Amount	Int/Float	130, 90	Y	Open	Rule and pattern
Price - Currency	String	§, долларов (= dollars)	Y	Closed	Dictionary
Price - Payment Period	String	mo, месяц (= maand)	Y	Closed	Dictionary
Service	String	shared хостинг (= hosting)	N	Open	Machine Learning
Service Details	String	Private data center	N	Open	Machine Learning
Support	String	24/7, technical	Y	Open	Machine Learning
Technical Specification	String	4GB RAM, Intel Xeon E5440	N	Open	Machine Learning
URL	String	<url>	Y	Open	Rule and pattern

TABLE 5.2: Sixteen identified attributes in BPH posts

5.5 Manual Annotation

For consistency in the annotation process, a set of guidelines was constructed. The guidelines were constructed based on the annotations of 15 random Russian and English posts that were verified by a domain expert. The annotation guidelines can be found in appendix C. Using the identified attributes as labels and the annotation guidelines, posts were manually annotated. GATE was used for the annotation. GATE is an offline tool that facilitates NLP tasks. The Russian posts were annotated using an English translation. To determine the correctness of the Russian annotation, a subset of 50 random annotated posts was assessed by a second annotator with knowledge of the Russian language. With respect to the labels in the 50 posts, 3.8% were changed, removed or added by the second annotator. Therefore, we can argue that the Russian posts were sufficiently annotated.

For more insight in the sixteen attributes, the class frequency distribution is visualized in figure 5.8. Noticeable is the large class imbalance for both English and Russian.

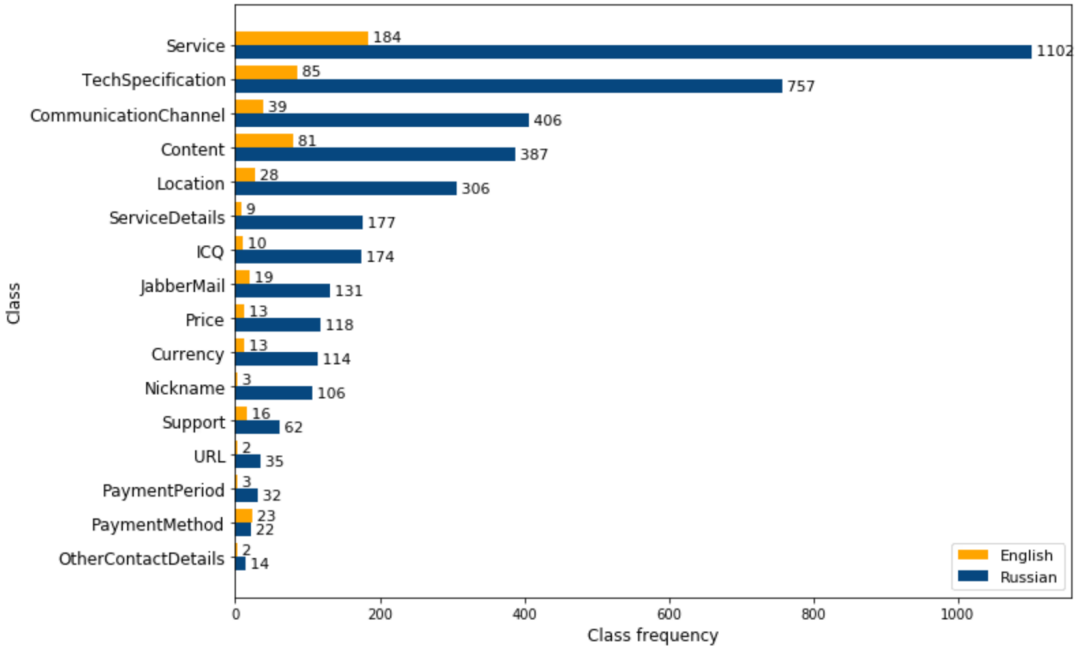


FIGURE 5.8: Class frequency distribution

Chapter 6

Methodology

This chapter describes the research methodology that was used to answer the research questions. For this study the domain was explored, relevant attributes were identified and the attribute extractors were developed and evaluated. Furthermore, transfer learning was applied in an attempt to improve the developed attribute extractors. In the following paragraphs, a general introduction of the methodology is given. Afterwards, in sections 6.1 to 6.9, each step in the methodology is explained in more detail. Table 6.1 provides an overview of the extraction techniques that were applied including the attributes they were applied to.

Attribute	B	D	R	C	H	L	T	E
URL	x		x	x	x	x	x	x
ICQ	x		x	x	x	x	x	x
Nickname	x			x	x	x	x	x
Communication Channel	x			x	x	x	x	x
Techical Specification	x			x	x	x	x	x
Service	x			x	x	x	x	x
Content	x			x	x	x	x	x
Location	x	x		x	x	x	x	x
Payment Period	x	x		x	x	x	x	x
Currency	x	x		x	x	x	x	x
Payment Method	x			x	x	x	x	x
Jabber Mail	x		x	x	x	x	x	x
Service Details	x			x	x	x	x	x
Other Contact Details	x			x	x	x	x	x
Support	x			x	x	x	x	x
Price	x		x	x	x	x	x	x

TABLE 6.1: Extraction techniques used per attribute

In chapter 5, the dataset construction was described followed by the data exploration and data preparation. Furthermore, the attributes in BPH posts were identified and used to manually annotate the data. Next, several techniques were used to perform attribute extraction aimed to compare the performance of these methods. These include a baseline extractor (B), dictionary-based extractor (D), rule- and pattern-based extractor (R), CRF extractor (C), hybrid CRF extractor (H) and a biLSTM-CRF extractor (L). For each technique, experiments were conducted to improve the performance.

Transfer learning was applied to the hybrid CRF extractor (H) and to the biLSTM-CRF extractor (L). The hybrid CRF extractor is based on a CRF, a traditional machine

learning model. Therefore, transfer learning was applied by including the results of pre-trained models as features. The biLSTM-CRF extractor is a neural network. Transfer learning was applied to the biSLTM-CRF by pre-training the embedding layer on a large corpus. This biSLTM-CRF extractor resulted in model T.

Finally, an ensemble extractor (E) was developed that combines for each attribute the best performing extractor.

6.1 Baseline Extractors

The baseline is a dictionary-based extraction method, i.e. it extracts a word, or multiple words, from text when it occurs in a dictionary. For this baseline, sixteen dictionaries were created, one for each attribute, and filled with the attributes in the labelled training dataset excluding duplicate values. For example, the location dictionary consists of all location attributes in the training dataset. The baseline extracts a word from unseen posts in the test dataset when it occurs in one of these dictionaries and assigns the corresponding class label, e.g. location if it occurs in the location dictionary.

The baseline has two main limitations. First of all, the baseline allows for words to be extracted more than once when it matches with multiple values in the dictionaries. For example, 'dedicated server' can be extracted as 'server' and 'dedicated server' if both occur in a dictionary. This is referred to as overlapping attributes. Secondly, true attributes are not extracted due to case sensitivity. To overcome these limitations, three adapted versions of the baseline model were tested. The first adapted version performs overlap removal to avoid the extraction of overlapping attributes. First, the overlapping attributes are detected based on the location of the words in a post. Subsequently, all except the largest attribute are discarded to ensure all possible relevant information is extracted. The second adapted version performs lowercasing of the posts as a pre-processing step. The third adapted version performs both lowercasing and overlap removal.

Another baseline extractor was developed and tested, referred to as the aggregated baseline extractor. This extractor was developed to test how good the original baseline extractor is at distinguishing between the sixteen attributes, i.e. the classes. For this purpose, the sixteen dictionaries are aggregated into one flat dictionary that discards all class information as depicted in figure 6.1. Using the aggregated dictionary, the model extracts words that occur in the dictionary, but it assigns all extracted words the same class label. It can be argued that, in case the aggregated baseline performs better compared to the original baseline, this indicates that the original baseline has difficulty assigning the correct class labels, e.g. due to the fact that attribute instances might be present in more than one attribute specific dictionary due to ambiguity.

Attribute	Dictionary
URL	url1, url2, url3
Currency	currency1, currency2

↓ *Aggregate*

Aggregated dictionary	url1, url2, url3, currency1, currency2

FIGURE 6.1: Aggregated dictionary created from attribute specific dictionaries

6.2 Dictionary Extractors

This section describes the dictionary-based attribute extractors for the closed attributes location, currency and payment period.

In contrast to the baseline, the dictionaries are based on external sources instead of annotated training data. For each attribute, several dictionaries were successively developed and evaluated to improve the performance. Similar to the baseline, the impact of lowercasing and overlap removal was tested. An exception is the payment period attribute, since the dictionary values are lowercased by default only overlap removal was tested.

To prevent false positives, regular expressions were used. The regular expression for location and currency extraction scans for matches that are not preceded or followed by a Latin or Cyrillic character. The regular expression for payment period scans the posts for periods that are preceded by a digit in the same sentence and not followed by a Latin or Cyrillic character.

Location extraction:

The first ('Naive') dictionary is based on the continents, countries and cities in the complete GeoNames data dump. GeoNames[1] is a database of known locations in multiple languages including English and Russian. For each location we included the full name, two-character code (ISO-3166), all alternative names and the iana timezone that also includes continent and city information. Furthermore, we added the locations 'world' and 'worldwide'. The final dictionary was obtained by removing duplicates and all locations that did not contain at least one character in the Latin or Cyrillic script.

The second ('Restrictive') dictionary is similar to the 'Naive' dictionary, but without the two-character codes and other locations with three characters or fewer.

The third ('Most Restrictive') dictionary is based on the country information from the GeoNames API. The benefit of using the API instead of the full data dump is that many uncommon locations that cause false positives are avoided. For each country we included the name, continent, and capital. To this we added the corresponding continents and alternative country names in Cyrillic script from the Geonames data

[1] https://www.geonames.org/ - a geographical database consisting of countries and other geolocations in i.a. English and Russian

dump. Furthermore, we added the locations 'world' and 'worldwide'.

Currency extraction:

The first dictionary ('Crypto and Regular') consists of the names and abbreviations of cryptocurrencies from Coinmarketcap[2]. Coinmarketcap is a webpage that reports the price of popular and obscure cryptocurrencies. The dictionary was extended with the ISO 4217 regular currencies including their abbreviation[3], the Russian machine translation of the regular currencies and the currency symbols[4].

The second dictionary ('Regular') is similar to the first dictionary, but without the cryptocurrency names and abbreviations from Coinmarketcap.

The third dictionary ('Restricted Regular') is similar to the second dictionary, but without the alphabetic, single character currency symbols. The non-alphabetic symbols were kept.

Payment period extraction:

The first dictionary ('Naive Periods') consists of commonly used time periods up to a year, ['second', 'minute', 'hour', 'day', 'week', 'month', 'year'], including their other forms, such as 'weekly', 'min', 'yr'. The dictionary was extended with Russian machine translations of the periods.

The second dictionary ('Realistic Periods') is similar to the first dictionary, but without periods that are uncommon for a payment period. These are the periods 'second', 'minute' and 'hour' and their other forms and translations.

6.3 Rule and Pattern Extractors

This section describes the rule- and pattern-based attribute extractors for the regular set attributes, i.e. ICQ, jabber mail and URL, and complex pattern attribute price(amount). For the regular set attributes, case insensitive regular expressions were developed to capture their distinctive internal structure. Another method is required to extract price values, since these are commonly integers and therefore lack a distinctive internal structure. Instead, structure in the word context was used to extract prices.

ICQ:
ICQ consists of a sequence of five or more digits, optionally interrupted by dashes or spaces. To prevent matching colour codes in BB-code tags, the regular expression was defined as follows:

```
\d+(\d(-| )?){3,}\d+(?!\d*\])
```

Jabber Mail:
The regular expression for jabber mail addresses is based on a naive structure of

[2]https://coinmarketcap.com/all/views/all/ - Coinmarketcap tracks the price of popular and obscure cryptocurrencies
[3]https://en.wikipedia.org/wiki/ISO_4217
[4]https://www.xe.com/symbols.php

mail addresses consisting of any character except newlines and whitespaces, interrupted by one at-sign and a dot symbol. Based on deviations of addresses written in BPH posts, the regular expression was adapted to a sequence of any characters with the exception of white spaces, newlines and square brackets, interrupted by one at-sign and a dot, either the symbol or written version, followed by Latin or Cyrillic characters:

```
[^\n\s@\]\[]+@[^\n\s@\[\]]+(\.|dot)[\u0400-\u04FFa-z]+
```

URL:
The regular expression for URL attributes is based on the URL standards described in the RFC-3986 standard[5]. However, since URLs in the posts do not have to comply to the official URL characters and structure, several adaptations were necessary. This resulted in the following regular expression:

```
([a-z]+:\/\/?(www\.)?|www\.)[\w\d\-\._~=;,\+\*\)\(\'&\$!@#\?\/:]+
\.[a-z]+[\w\d\-\._~=;,\+\*\)\(\'&\$!@#\?\/:%]*
```

Several URLs did not contain protocol information as well as no 'www'. Therefore a more relaxed regular expression was also tested:

```
([a-z]+:\/\/?(www\.)?|www\.)?[\w\d\-\._~=;,\+\*\)\(\'&\$!@#\?\/:]+
\.[a-z]+[\w\d\-\._~=;,\+\*\)\(\'&\$!@#\?\/:%]*
```

Price:
The regular expression for price attributes is based on structure in word context that is indicative for prices. A main indicator or a price is the occurrence of a currency before or after an integer or float. This was incorporated in the first regular expression. For this we used the currency dictionary from section 6.2 that resulted in the best performance. The expression is as follows:

```
'(('+LB_CUR+')\d+(\.\d+)?|
\d+(\.\d+)?(?= ?('+str('|'.join(x for x in CUR))+')))'
```

`LB_CUR` consists of a collection of look behind actions. These actions look for a known currency that precedes the price directly or with a whitespace in between. Separate look behind actions were required due to a fixed-length restriction. As an indication the two first actions are as follows: `((?<=\$)|(?<=\$\s)|...)`. The look forward action does not have a fixed-length restriction, therefore the list of known currencies, `CUR`, was used for the look forward action was as follows: `(?= ?(\$|...))`.

It is also common for a price to be followed by a payment period within the same sentence. The second regular expression incorporates this indicator by extending the first regular expression. The last two rows of the extended regular expression shown below contain the extension. `PER` refers to the payment period dictionary from section 6.2 that resulted in the best performance. To prevent false positives, several restrictions were added. So should the period start within 10 characters from the integer or float and should the payment period not be directly preceded or followed by a Latin or Cyrillic character.

```
'(('+LB_CUR+')\d+(\.\d+)?|
\d+(\.\d+)?(?= ?('+str('|'.join(x for x in CUR))+'))|
\d+(\.\d+)?(?=.{0,10}[^\u0400-\u04FFA-Z]('+str('|'.join(x for x in
PER))+')[^\u0400-\u04FFA-Z]))'
```

[5]RFC-3986 standard: http://www.ietf.org/rfc/rfc3986.txt

6.4 CRF Extractors

Sequence labelling with traditional machine learning algorithms, such as CRF, can be performed with a wide variety of configurations. The configurations include the algorithm itself, its hyper parameters, text pre-processing, token representation into feature vectors and the definition of a document, e.g. a sequence of tokens in a post or in a sentence.

Several CRF extractors were developed for this study. The first, the initial CRF extractor, is a basic CRF extractor that is explained in section 6.4.1. Afterwards, a series of experiments was conducted aimed to improve the performance of the initial CRF extractor by discovering good configurations for this particular application. The experiments are explained in section 6.4.2. The final CRF extractor was constructed based on the outcome of all experiments and is explained in the results section 7.6. This process was performed separately for English and Russian.

6.4.1 Initial CRF Extractor

The initial CRF extractor makes use of the discriminative linear chain CRF (LC-CRF) from the sklearn-crfsuite[6]. The LC-CRF, described in section 4.1.4, and makes of the preceding predicted class label and the feature vector of the current token, i.e. transition and state feature functions respectively. The default hyper-parameter values were used with the L-BFGS training algorithm and L1 and L2 regularization.

Data Preparation

Posts were split into tokens using a tokenizer. For this purpose, a tokenizer was developed that splits on pre-determined delimiters in order to meet several main requirements. The first requirement was that no token contained two separate attribute instances. The second requirement was to keep the BB-codes intact. Each token was then paired with their BIO encoded label. Furthermore, for the initial CRF extractor, no data pre-processing was performed and the posts were not split into smaller paragraphs or sentences. Instead the post sequence definition was used.

Feature Engineering

Feature engineering is the process of representing tokens as feature vectors that are used as input for the CRF extractor. For the initial CRF extractor, each token t_i was represented with features from the token itself and from the neighbouring tokens, t_{i-1} and t_{i+1}, resulting in a total of 27 features per feature vector. Tokens from the neighbouring tokens are referred to as context features.

A basic set of word-level features, list lookup features and document and corpus features were used that are commonly used for NER [25]. The word-level features used are the token, three-character prefix, three-character suffix and three boolean features representing the presence of a capital character, digit and any special character in the token respectively. The list lookup feature used is a boolean feature representing the presence of the token in a list of English and Russian stopwords from NLTK. The document and corpus features used are the token index in the sequence and token frequency in the corpus.

[6] https://sklearn-crfsuite.readthedocs.io/en/latest/index.html

6.4.2 Experiments

The following paragraphs describe the experiments that were conducted to improve the performance of the initial CRF extractor. The first experiment compared our initial CRF extractor, a sequence model, with a regular classifier. The second experiment compared the performance impact of several sequence definitions. The third experiment tested the impact of performing hyper-parameter tuning. The fourth experiment looked at the impact of extending the feature vectors. The fifth experiment focussed on the impact of data pre-processing steps. The outcome of each experiment was used as input for the next experiment.

Experiment 1: Machine Learning Algorithms

The first experiment focussed on comparing our initial CRF extractor, a sequence model, against a regular classifier. For the regular classifier we used the discriminative MEM classifier from NLTK[7] with default hyper-parameters. Everything else was kept the same for both extractors.

Experiment 2: Sequence Definition

The second experiment compared three sequence definitions: i) post-level, ii) paragraph-level and iii) sentence-level sequences. For the post-level sequence, each post is regarded as one sequence. This was used for the initial CRF extractor. For the paragraph-level sequence, each paragraph in the posts is regarded as one sequence. For the sentence-level sequence, each sentence in the posts is regarded as one sequence. The number of posts, paragraphs and sentences in the English and Russian datasets were shown in section 5.3.2.

Experiment 3: Hyper-parameter Tuning

The third experiment tested the performance impact of hyper-parameter tuning of the CRF extractor. The GridSearchCV hyper-parameter optimizer from scikit-learn[8] was used. It performs an exhaustive search through a set of predetermined parameter values. The tuning consisted of two phases. In the first phase, the discrete valued parameters were tuned. Here all values for the discrete parameters were tested in combination with the default valued continuous parameters. In the second phase, the continuous valued parameters were tuned. Here three up to five values were tested for each continuous parameter including the default value. The discrete parameters were set to the tuned parameter values from phase one. The set of discrete and continuous parameters and their pre-determined parameter values can be found in appendix A.

Experiment 4: Feature Engineering

The fourth experiment tested the impact of extending the features in the feature vector. The process consisted of feature development and feature selection.

For feature development, the set of word-level features, list lookup features and document and corpus features was extended. This resulted in 82 features, not including context features. The extended feature set can be found in appendix B. Several of

[7] https://www.nltk.org/_modules/nltk/classify/maxent.html
[8] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

the new features were dependent on the training data. An example is the boolean feature 'isKnownService' that is true in case the token matches a service attribute in the training data. It was ensured that the test data was not used to generate this feature.

Feature selection was performed to select only the features that contain information that is beneficial for the attribute extraction. For this purpose, the CRF extractor was trained and tested iteratively. In each iteration, one of the new features was added to the initial feature vector and removed afterwards. Features were said to be beneficial if they resulted in an improved performance with respect to using the tuned CRF with the initial feature vector from the previous experiment. Feature selection was performed for features representing the current token as well context features of the 4 neighbouring tokens, i.e. a window of five tokens. However, context features that were an exact replica of the current token feature, e.g. the sequenceLength feature, were not included.

Experiment 5: Data Pre-processing

The fifth experiment tested the impact of several data pre-processing steps. These are the text normalization steps lowercasing, stemming, punctuation removal and stopword removal. Furthermore, we evaluated the impact of removing BB-codes from the posts as a noise-removal step. The stemmers used are the Porter and Snowball stemmers for English and Russian respectively. Both are available in the stem package of NLTK[9]. The punctuation removal step removed the punctuation tokens (`.?!,;:-()`) that where already split into separate tokens by the tokenizer. For stopword removal, tokens that occur in the English or Russian NLTK stopword lists were removed. Each BB-code was also tokenized as one token. BB-code removal could therefore be performed by removing tokens matching the regular expression for BB-codes: `\[/?\w+(="?#?\w+"?)?\]` .

6.5 Hybrid CRF Extractors

Besides the individual dictionary-, rule- and machine learning based methods for attribute extraction, we explored the performance impact of combining the methods into a hybrid extractor. The hybrid CRF extractor consists of the final CRF extractor described in section 6.4, but with additional features based on external sources. These features are referred to as hybrid features. The impact of each hybrid feature was tested individually. The features that resulted in a performance improvement were added to the feature vector of the final hybrid CRF extractor. This was performed for English and Russian separately. The feature additions can be split in three categories.

The first two categories consists of new features that incorporate the output of best performing dictionary extractors and rule based extractors described in section 6.2 and 6.3 respectively. These are referred to as dictionary features and rule features. Dictionary features are boolean features indicating whether the token consist in a list of known attributes, e.g. isInGeoNamesLocations. The rule features are boolean features indicating whether the token is part of an attribute extracted by a regular

[9] https://www.nltk.org/api/nltk.stem.html

expression, e.g. isPartOfURL.

The third category consists of features that incorporate the output of pre-trained models. This is transfer learning for traditional machine learning models as described in section 3.2.1. These features are therefore referred to as transfer learning features. The first feature captures the part-of-speech tag of the token. For this we used the English and Russian version of the part-of-speech tagger called TreeTagger[10]. The second feature addition resulted in a new feature for each value in a 100 dimensional word embedding of the token. The third feature addition consisted of the 1, 5, 10, 20 or 40 most similar words of a token based on learned word embeddings. The word embedding model was trained on all posts of the DNM, irrespective of category and language. For this purpose, word2vec from the Gensim Python library[11] was used. Word embeddings are words mapped to vectors of real numbers. These mappings can be learned using word2vec, a shallow neural network that receives as input a large corpus and returns a vector space with a vector representing each word. As a result, word vectors that are close to each other in the vector space represent words that occur in a similar text context in the input corpus. A word embedding model was trained for the English and Russian hybrid CRF extractors separately to ensure the same data preparation and pre-processing methodology was used as that of the final English and Russian CRF extractors from the previous section.

6.6 biLSTM-CRF Extractors

This section describes how the biLSTM-CRF neural network was used to perform machine learning based attribute extraction. The biLSTM-CRF extractor was developed to compare its performance with that of the traditional CRF extractors described in the previous two sections.

Several biLSTM-CRF extractors were developed for this study. The first, the initial biLSTM-CRF extractor, is explained in section 6.6.1. Afterwards, experiments were conducted aimed to improve the performance of the initial biLSTM-CRF extractor. The experiments are explained in section 6.6.2. The final biLSTM-CRF extractor was constructed based on the outcomes of all experiments and is explained in the results section 7.8. This process was performed separately for English and Russian.

6.6.1 Initial biLSTM-CRF

Model Input

For good comparison, the data preparation was kept identical to that of the CRF described in section 6.4.1. Next, the data was formatted to make it compatible as model input. For this purpose, the sequences of tokens and labels were transformed to vectors of integers. These integers are the unique IDs of tokens and labels respectively. To obtain equal length input vectors, they were padded with ID 0, the padding token. These zero values were ignored by the model by setting the mask_zero parameter in the embedding layer to True.

[10]Part-of-speech tagger TreeTagger: https://treetaggerwrapper.readthedocs.io
[11]Source Gensim Python library: https://radimrehurek.com/gensim/

Architecture

The initial biLSTM-CRF consists of a word embedding layer, with randomly initialized weights, followed by a bidirectional LSTM and a CRF layer. In order to obtain a good performance, the right hyper-parameters should be used [41]. The hyper-parameters described in the next paragraph are based on the optimal hyper-parameters for a biLSTM-CRF for sequence labelling tasks, as found by Reimers et al. [41].

Details of the model architecture including hyper-parameters are as follows: The word embeddings, with a dimension of 100, are used as input for the forward and backward LSTM layers. For each LSTM layer the number of recurrent units is 100 and both dropout and recurrent dropout are set to 0.5 to prevent overfitting. The output of the LSTM layers is then concatenated and used as input for the CRF. To ensure the output of the bidirectional LSTM is sequential, i.e. an output for each token in a sequence, the return_sequences parameter is set to True. To train the biLSTM-CRF, the Adam with Nesterov momentum optimizer (Nadam) is used with the default learning rate of 0.002 in combination with a batch size of 8 and 25 epochs. For the initial biLSTM-CRF, the sentence sequence definition was used.

6.6.2 Experiments

The following paragraphs describe the experiments that were conducted to improve the performance of the initial biLSTM-CRF extractor. The first experiment tested the impact of the sequence definitions. The second experiment tested the impact of changing hyper-parameters and architecture. The third experiment tested the impact of several data pre-processing steps. All experiments were conducted in isolation, i.e. one change with respect to the initial biLSTM-CRF extractor.

Experiment 1: Sequence Definition

The first experiment compared the post-level, paragraph-level and sentence-level sequence definitions.

Experiment 2: Hyper-parameters and Architecture

The second experiment explored the performance impact of adapting the model hyper-parameters and architecture. Changing certain hyper-parameters has a larger performance impact compared to changing others [41]. Based on this finding experiments were conducted to find good values for the hyper-parameters batch size, epochs, dropout and embedding dimension.

For batch size, we experimented with sizes 8 and 32. Even though a batch size closer to 32 was found to be optimal for named entity recognition and related tasks, a lower batch size of 8 is more suited for small datasets, since it results in more robust models [41]. The epoch sizes tested are 25, 100 and 200. The dropout values tested are 0.15, 0.25 and 0.5. Furthermore, the impact of using only regular dropout and only recurrent dropout was explored. Lastly, the embedding dimensions tested are 100, 200 and 300.

For the architecture, we explored the impact of using a single (forward) LSTM instead of the bidirectional LSTM. Furthermore, we tested the impact of using both

word and character embeddings compared to using solely word embeddings. The character embeddings have a dimension of 100. When using both embedding types, separate word embedding and character embedding layers are trained and the outcomes are concatenated. The word embedding contains information about the word context whereas the character embedding contains information about orthographic similarities, i.e. what the word looks like [7].

Experiment 3: Data Pre-processing

The third experiment compared several pre-processing steps. These are lowercasing, stemming, BB-code removal, punctuation removal and stopword removal. The same techniques and resources were used as described in the CRF pre-processing experiment in section 6.4.2.

6.7 Transfer Learning with the biLSTM-CRF Extractors

This section describes how transfer learning by fine-tuning was applied in an attempt to improve the performance of the biLSTM-CRF extractor. Several experiments were conducted to test the applicability of transfer learning. The first experiment, described in section 6.7.1, tested the impact of applying transfer learning on the word embedding layer versus both the word and character embedding layers. The second experiment, described in section 6.7.2, tested the impact of using various source datasets for pre-training. This process was performed separately for English and Russian

6.7.1 Experiment 1: Transferring Word Embeddings vs. Word and Character Embeddings

The first experiment compared the impact of applying transfer learning on the word embedding layer versus both the word and character embedding layers.

First, transfer learning was applied on the word embedding layer. The remaining layers, including the character embedding layer, were trained from scratch. For this purpose, an embedding model was pre-trained on a large source dataset. The source dataset consisted of all posts of the DNM, irrespective of category and language. Then, instead of random weight initialization of the word embedding layer of our target biLSTM-CRF model, the weights were initialized according to the learned weights in the pre-trained embedding model. Lastly, the target model was trained on the target dataset as you would without any transfer learning. This fine-tuned the weights for our target attribute extraction task.

To pre-train the word embedding model on the source dataset, word2vec from the Gensim Python library [12] was used. The dimensionality of the embeddings was set to the default value of 100. Before pre-training, preparation and pre-processing of the source data was performed equal to the description given in previous section 6.6 and based on the outcome of the pre-processing experiment.

For the second part of the experiment, we applied transfer learning on both the word and character embedding layers. The remaining layers were trained from scratch.

[12] Source Gensim Python library: https://radimrehurek.com/gensim/

For this purpose, the same pre-trained word embedding model was used. To pre-train the character embedding model, the same source dataset and methodology was used as described in the previous two paragraphs. The difference was that, instead of using sequences of words as model input, we used sequences of characters.

6.7.2 Experiment 2: Comparing Source Datasets

As described in section 3.2.2 of the related work, the suggested approach is to select a source dataset with a large vocabulary size and a low out-of-vocabulary metric. Therefore, the source dataset consisting of all posts of the DNM was used for the first transfer learning experiment. However, the semantic similarity is also of importance. Therefore, in this second experiment, we explored several other source datasets.

The second source dataset consists of the language filtered DNM posts, i.e. all Russian and English posts for the Russian and English biLSTM-CRF extractor respectively. The third source dataset is a copy of the second dataset, but only contains posts from the hosting and related services category. The fourth source dataset consists of the language filtered posts with a hosting category from multiple DNMs including our original DNM.

6.8 Ensemble Extractors

An ensemble extractor was developed that combines for each attribute the best performing extractor. The extractors considered are the dictionary-based extractor, rule-based extractor, CRF extractor, hybrid CRF extractor, biLSTM-CRF extractor and biLSTM-CRF extractor developed with transfer learning. For this extractor we used the sentence sequence definition. The ensemble extractor was developed as follows: for each attribute, the best performing extractor is selected based on the F2-score for that attribute. The predicted labels obtained by these selected extractors are then combined. Since two models can assign different labels to the same token, a sequential approach is used to give priority to labels of attributes with a higher F2-score. This begins by adding the labels of the attribute with the highest F2-score and ends with adding the labels of the attribute with the lowest F2-score.

6.9 Training and Testing

Due to the small dataset, five-fold cross-validation was applied to train and test the machine learning models and the baseline model. This resulted in a more accurate model performance estimate. With five-fold cross-validation, the dataset is randomly split in five parts. During training, one part is selected for testing and the remaining four are used to train the model. This is performed five times, each time with a different part for testing. The split in five folds, i.e. how the input data and classes are distributed over the five folds, influences the model performance. Therefore, generating five folds for each model resulted in varying performance influences that were not caused by the models themselves. To avoid this and better compare the performance of the models, the five folds were generated once and used to train and test all models.

Chapter 7

Experiments and Results

This chapter evaluates the results of all attribute extractors including experiments and transfer learning attempts. Section 7.1 describes the experimental setup. An overview of the main findings including a comparison of the extraction techniques is provided in section 7.2. Sections 7.3 to 7.9 provide for each extraction technique and transfer learning attempt an in depth analysis of the results.

7.1 Experimental Setup

This section describes the experimental setup including a recap of the data and the method and performance metrics used to evaluate the attribute extractors.

The data used consists of the English and Russian datasets as described in chapter 5. The datasets consist of 62 and 470 BPH posts respectively. The following sixteen attributes can be found in the data: URL, ICQ, nickname, communication channel, technical specification, service, content, location, payment period, currency, payment method, jabber mail, service details, other contact details, support and price. BIO encoding was used to annotate the data. This resulted in 33 classes, two per attribute in addition to the 'O' class for non-attribute tokens.

To evaluate the attribute extractors, the MUC evaluation method was used as described in section 4.3. This method allows for partial matches where either the correct class is predicted or the correct boundaries of the attribute are found. The inclusion of partial matches in addition to exact matches is of importance due to the fact that partial matches can also provide relevant information for a police investigation. For example, the extraction of 'hosting' is still relevant even though the true boundaries contain 'bulletproof hosting'. However, the exact matches receive more weight compared to partial matches.

The performance metrics used are precision, recall and F2-score. The exact formula's used to calculate the performance of an extractor can be found in section 4.3. The F2-score was used instead of the F1-score, since extracting all attribute instances (recall) is of greater importance than extracting solely correct attribute instances (precision). To compare the machine learning based attribute extractors and the impact of transfer learning, we provided the micro-averaged ($F2_{Mi}$), macro-averaged ($F2_{Ma}$) and weighted macro-averaged ($F2_{W}$) F2-scores as well as the micro-averaged precision (P_{Mi}) and recall (R_{Mi}). However, due to class imbalance as described in section 5.5, we evaluated the attribute extractors based on the $F2_{Mi}$.

Furthermore, the development of attribute extractors and the application of transfer learning was performed for English and Russian separately.

7.2 Result Overview

In this study, several attribute extraction techniques were used and compared to find the best performing extractor per attribute. The extractors developed include a baseline extractor (B), dictionary-based extractor (D), rule- and pattern-based extractor (R), CRF extractor (C), hybrid CRF extractor (H), biLSTM-CRF extractor (L), a biLSTM-CRF extractor developed with transfer learning (T) and an ensemble extractor (E). In this section, the main findings of this study are presented. An overview of the best obtained F2-scores is shown in table 7.1.

Dictionary-based and Rule- and Pattern-based Extraction

For the location, payment period and currency attributes, dictionary-based extraction (D) was performed. This resulted in the best performance for the location and payment period attributes for English. However, machine learning approaches outperformed the dictionary-based approach for the other attributes.

For the URL, ICQ, jabber mail and price attributes, rule- and pattern-based extraction (R) was performed. For both English and Russian, this resulted in the best performance for all four attributes. The exception is price attribute extraction for Russian. This was best performed with machine learning.

Machine Learning Based Extraction

For both English and Russian, it was found that the hybrid extractor (H), with a $F2_{Mi}$-score of 0.825 and 0.877 respectively, outperformed the CRF extractor (C) as well as the biLSTM-CRF extractor (L). However, the performance of machine learning based extractors is attribute dependent. Therefore, some attributes were extracted with a higher F2-score by the biLSTM-CRF (L) or CRF (C) extractors instead of the hybrid extractor (H).

Transfer Learning

For both English and Russian, training the biLSTM-CRF with transfer learning (T) resulted in an improved performance compared to training from scratch (L). However, this requires a source dataset that consists of a sufficient amount of language filtered posts about a related topic, e.g. Russian hosting posts for Russian. Furthermore, for some attributes transfer learning resulted in a worsened F2-score.

Ensemble Extractor

For English, the ensemble extractor (E) outperformed all other machine learning based extractors. For Russian, the ensemble extractor (E) outperformed the CRF (C), hybrid (H) and biLSTM-CRF (L) extractors. However, it was not able to outperform the biLSTM-CRF developed with transfer learning (T).

Conclusion

For English, the ensemble extractor (E) resulted in the best performance with a $F2_{Mi}$-score of 0.859. For Russian, this was not the case as the biLSTM-CRF developed with transfer learning (T) resulted in the best performance with a $F2_{Mi}$-score of 0.886.

	Attribute $_{(class frequency)}$	B	D	R	C	H	L	T	E
English	URL (2)	0.000		1.000	0.000	0.000	0.000	0.000	1.000
	ICQ (10)	0.533		1.000	0.880	0.816	0.913	0.880	1.000
	Nickname (3)	0.000			0.000	0.000	0.000	0.000	0.000
	Comm. Channel (39)	0.777			0.905	0.930	0.974	0.954	0.974
	Tech. Spec. (85)	0.240			0.712	0.689	0.625	0.640	0.714
	Service (184)	0.832			0.888	0.890	0.911	0.894	0.916
	Content (81)	0.848			0.787	0.803	0.881	0.813	0.881
	Location (28)	0.729	0.899		0.603	0.638	0.496	0.472	0.870
	Payment Period (3)	0.250	1.000		0.882	0.833	0.714	0.714	1.000
	Currency (13)	0.467	0.747		0.794	0.847	0.917	0.917	0.917
	Payment Method (23)	0.290			0.636	0.625	0.467	0.514	0.625
	Jabber Mail (19)	0.493		1.000	1.000	0.916	0.964	1.000	1.000
	Service Details (9)	0.000			0.000	0.000	0.000	0.000	0.000
	Other Contact D. (2)	0.400			1.000	1.000	1.000	1.000	1.000
	Support (16)	0.467			0.577	0.570	0.570	0.617	0.625
	Price (13)	0.150		0.985	0.703	0.703	0.738	0.794	0.985
	Best F2$_{Mi}$	***0.618***			***0.820***	***0.825***	***0.804***	***0.806***	***0.859***
Russian	URL (35)	0.147		0.783	0.729	0.738	0.711	0.675	0.777
	ICQ (174)	0.819		0.973	0.945	0.953	0.934	0.944	0.977
	Nickname (106)	0.645			0.684	0.723	0.613	0.632	0.662
	Comm. Channel (406)	0.714			0.915	0.907	0.921	0.918	0.922
	Tech. Spec. (757)	0.579			0.807	0.818	0.813	0.837	0.827
	Service (1102)	0.862			0.939	0.932	0.927	0.932	0.935
	Content (387)	0.530			0.754	0.771	0.797	0.793	0.780
	Location (306)	0.818	0.528		0.841	0.877	0.860	0.870	0.860
	Payment Period (32)	0.306	0.886		0.774	0.774	0.890	0.849	0.890
	Currency (114)	0.322	0.766		0.941	0.954	0.963	0.943	0.963
	Payment Method (22)	0.480			0.400	0.571	0.337	0.347	0.365
	Jabber Mail (131)	0.850		0.989	0.974	0.979	0.983	0.980	0.989
	Service Details (177)	0.371			0.677	0.680	0.691	0.693	0.693
	Other Contact D. (14)	0.622			0.882	0.882	0.797	0.857	0.882
	Support (62)	0.693			0.787	0.776	0.697	0.771	0.795
	Price (118)	0.248		0.908	0.934	0.944	0.920	0.905	0.949
	Best F2$_{Mi}$	***0.637***			***0.870***	***0.877***	***0.872***	***0.886***	***0.884***

TABLE 7.1: Attribution extraction result overview - The best F2-scores

7.3 Baseline Extractors

This section shows and evaluates the baseline extractor results. Several baselines were developed. These are the original baseline and three improved baselines that include lowercasing and overlap removal. Furthermore, an aggregated baseline extractor was developed to test how good the original baseline is at distinguishing between the sixteen classes. First, the result of the original baseline and the impact of lowercasing and overlap removal is evaluated. Afterwards, the results of the aggregated baseline is provided. The full set of performance scores of the baseline extractors including precision and recall can be found in appendix D.

The results of the original baseline extractor and the impact of lowercasing (L) and overlap removal (OR) are shown in table 7.2. With respect to the $F2_{Mi}$-score of the baseline extractor, overlap removal resulted in a large performance improvement for both English and Russian. However, lowercasing worsened the performance. This is arguably due to ambiguity caused by lowercasing that resulted in more attribute values that occur in two or more dictionaries. Interesting is that combining lowercasing and overlap removal resulted in the best performance for the original Russian baseline extractor. This indicates that lowercasing resulted in more overlapping attributes and that overlap removal prevented this from worsening the performance. From the F2-scores of the attributes, it can be inferred that the impact of lowercasing and overlap removal is attribute and dataset dependent.

	English				Russian			
Attribute	Raw	L	OR	L+OR	Raw	L	OR	L+OR
Service	0.485	0.438	0.811	0.832	0.554	0.533	0.860	0.862
ServiceDetails	0.000	0.000	0.000	0.000	0.216	0.199	0.371	0.369
TechSpecification	0.240	0.190	0.240	0.201	0.457	0.408	0.530	0.579
Location	0.729	0.538	0.729	0.729	0.800	0.685	0.818	0.779
Price	0.150	0.150	0.150	0.150	0.091	0.091	0.245	0.248
Currency	0.267	0.467	0.267	0.467	0.316	0.277	0.322	0.319
PaymentPeriod	0.250	0.183	0.250	0.217	0.237	0.224	0.306	0.291
PaymentMethod	0.290	0.290	0.290	0.290	0.480	0.433	0.480	0.433
CommChannel	0.759	0.748	0.777	0.766	0.655	0.621	0.714	0.672
ICQ	0.533	0.533	0.533	0.533	0.819	0.819	0.819	0.819
Nickname	0.000	0.000	0.000	0.000	0.596	0.550	0.645	0.583
JabberMail	0.493	0.493	0.493	0.493	0.850	0.850	0.850	0.850
OtherContactD.	0.400	0.400	0.400	0.400	0.622	0.605	0.622	0.605
Content	0.818	0.828	0.818	0.848	0.416	0.425	0.509	0.530
Support	0.270	0.382	0.460	0.467	0.653	0.682	0.663	0.693
URL	0.000	0.000	0.000	0.000	0.133	0.133	0.147	0.147
$F2_{Mi}$	0.485	0.447	0.618	0.612	0.469	0.447	0.634	0.637

TABLE 7.2: F2-scores of the original baseline extractor and the impact of lowercasing (L) and overlap removal (OR)

The aggregated baseline results together with the original baseline results are shown in table 7.3. Comparing the best $F2_{Mi}$-scores, the aggregated baseline outperforms the original baseline with 0.038 and 0.082 $F2_{Mi}$-score for English and Russian respectively. From this we can infer that the original baseline is able to distinguish between

the sixteen classes and that the aggregated model does not provide large additional benefits.

	English				Russian			
Baseline	Raw	L	OR	L+OR	Raw	L	OR	L+OR
Aggregated	0.502	0.472	0.651	0.656	0.485	0.477	0.705	0.719
Original	0.485	0.447	0.618	0.612	0.469	0.447	0.634	0.637

TABLE 7.3: $F2_{Mi}$-scores of the aggregated and original baseline extractor

7.4 Dictionary Extractors

In this section, the results of the dictionary-based attribute extractors are shown and evaluated for the attributes location, currency and payment period.

Location:

Three dictionaries were compared for location extraction. These are the 'Naive', 'Restrictive' and 'Most Restrictive' dictionaries. Furthermore, the impact of lowercasing the text and performing overlap removal was explored. The results are presented in table 7.4.

The 'Naive' dictionary resulted in the highest recall for both English and Russian due to the fact that many locations are included in the dictionary. However, for that same reason, many false positives were obtained. This resulted in a low precision and F2-score. The reason is that location names can be ambiguous and match other English or Russian words and abbreviations. An example is Many, a town in Louisiana. Furthermore, the large number of locations with three or fewer characters worsened the precision.

The 'Restrictive' dictionary, with the locations of thee or fewer characters removed, resulted in a slight improvement of the precision, but a worsened recall. This holds for both English and Russian.

The 'Most Restrictive' dictionary resulted in the best performance for both English and Russian with a F2-score of 0.899 and 0.528 respectively. A large increase of the precision was measured with respect to the 'Naive' and 'Restrictive' dictionary. However, the recall was worsened with respect to the 'Naive' dictionary, but increased with respect to the 'Restrictive' dictionary. The low recall of the Russian location extractor indicates that the GeoNames locations in Cyrillic script are not sufficient for this task. This is due to the variety in writing style and due to the numerous possible suffixes for one location. For example, Europe was written as Европе, Европа and Европы and not all versions are incorporated in GeoNames.

When using the 'Most Restrictive' dictionary, lowercasing increased the F2-score for both English and Russian whereas removing overlapping attributes did not have a performance impact.

			English			Russian		
Dictionary	L	OR	F2	P	R	F2	P	R
Naive			0.163	0.039	0.786	0.213	0.065	0.498
Naive	x		0.091	0.020	0.964	0.112	0.027	0.539
Naive		x	0.173	0.042	0.821	0.218	0.066	0.508
Naive	x	x	0.094	0.020	0.982	0.112	0.027	0.538
Restrictive			0.326	0.106	0.679	0.338	0.203	0.405
Restrictive	x		0.171	0.041	0.857	0.211	0.068	0.444
Restrictive		x	0.352	0.116	0.714	0.347	0.212	0.413
Restrictive	x	x	0.177	0.042	0.875	0.211	0.068	0.443
Most restrictive			0.870	0.923	0.857	0.510	0.976	0.456
Most restrictive	x		0.899	0.926	0.893	0.528	0.904	0.479
Most restrictive		x	0.870	0.923	0.857	0.510	0.976	0.456
Most restrictive	x	x	0.899	0.926	0.893	0.528	0.904	0.479

TABLE 7.4: The F2-score (F2), precision (P) and recall (R) for dictionary-based location extraction and the impact of lowercasing (L) and overlap removal (OR)

Currency:

Three dictionaries were compared for currency extraction. These are the 'Crypto and Regular', 'Regular' and 'Restricted Regular' dictionaries. Furthermore, the impact of lowercasing and overlap removal was explored. The results are presented in table 7.5

For the Russian currency extraction, the 'Crypto and Regular' dictionary resulted in a small recall improvement compared to the 'regular' and 'restricted regular' dictionaries. Cryptocurrencies are not commonly used to describe the price of BPH. However, the few cryptocurrencies used were captured by including cryptocurrencies in the dictionary. This resulted in the recall increase. However, the addition of cryptocurrencies also resulted in a low precision due to false positives that were mainly caused by ambiguous cryptocurrency abbreviations, e.g. Ethereum Classic (ETC). In the English posts no cryptocurrencies are mentioned. Therefore, the 'crypto and regular' dictionary only resulted in a worsened precision compared to the other dictionaries.

The 'Regular' dictionary, without cryptocurrencies, resulted in an improved precision, but also a slight reduction in recall for the Russian extractor. However, especially for the Russian extractor, the precision remained low due to false positives caused by alphabetic, single character currency symbols.

The 'Restricted Regular' dictionary, without alphabetic, single character currency symbols, resulted in the best performance for both English and Russian with a F2-score of 0.747 and 0.766 respectively.

It was found that overlap removal did not have a performance impact. Lowercasing improved the recall, but largely decreased the precision. Recall improved due to the fact that false negatives caused by unconventional usage of capital characters in currencies is avoided, e.g. writing 'usd' instead of 'USD'. However, the main reason

lowercasing decreased precision is the occurrence of ambiguous currency abbreviations, e.g. the Albanian lek (ALL), that cause more false positives when lowercased. Removing all abbreviations largely reduced the recall and was therefore not an option.

			English			Russian		
Currency dictionary	L	OR	F2	P	R	F2	P	R
Crypto and Regular			0.572	0.306	0.731	0.563	0.266	0.781
Crypto and Regular	x		0.194	0.047	0.885	0.449	0.154	0.860
Crypto and Regular		x	0.572	0.306	0.731	0.563	0.266	0.781
Crypto and Regular	x	x	0.194	0.047	0.885	0.449	0.154	0.860
Regular			0.731	0.731	0.731	0.704	0.537	0.763
Regular	x		0.728	0.426	0.885	0.660	0.354	0.842
Regular		x	0.731	0.731	0.731	0.704	0.537	0.763
Regular	x	x	0.728	0.426	0.885	0.660	0.354	0.842
Restricted Regular			0.731	0.731	0.731	0.766	0.777	0.763
Restricted Regular	x		0.747	0.460	0.885	0.720	0.455	0.842
Restricted Regular		x	0.731	0.731	0.731	0.766	0.777	0.763
Restricted Regular	x	x	0.747	0.460	0.885	0.720	0.455	0.842

TABLE 7.5: The F2-score (F2), precision (P) and recall (R) for dictionary-based currency extraction and the impact of lowercasing (L) and overlap removal (OR)

Payment Period:

Two dictionaries were compared for payment period extraction. These are the 'Naive Periods' and 'Realistic Periods' dictionaries. Furthermore, the impact of overlap removal was explored. The results are presented in table 7.6

For the English payment period extraction, both dictionaries resulted in a F2-score of 1.000. For the Russian payment period extraction, the 'Realistic Periods' dictionary resulted in the best performance with a F2-score of 0.886. The 'Naive Periods' dictionary resulted in a lower performance. The reason is that the periods that are not commonly used as payment period caused many false positives. The recall of the 'Realistic Periods' dictionary slightly decreased with respect to the 'Naive Periods' dictionary due to the fact that uncommon periods do occasional occur as payment period. For English as well as Russian, overlap removal did not have an impact on the performance.

		English			Russian		
Period dictionary	OR	F2	P	R	F2	P	R
Naive Periods		1.000	1.000	1.000	0.808	0.457	1.000
Naive Periods	x	1.000	1.000	1.000	0.808	0.457	1.000
Realistic Periods		1.000	1.000	1.000	0.886	0.660	0.969
Realistic Periods	x	1.000	1.000	1.000	0.886	0.660	0.969

TABLE 7.6: The F2-score (F2), precision (P) and recall (R) for dictionary-based payment period extraction and the impact of overlap removal (OR)

7.5 Rule and Pattern Extractors

In this section, the results of the rule- and pattern-based attribute extractors are shown and evaluated for the attributes ICQ, jabber mail, URL, and price. The performance of all four attributes is shown in table 7.7.

	English			Russian		
Rule/Pattern	F2	P	R	F2	P	R
ICQ	1.000	1.000	1.000	0.973	0.934	0.983
Jabber Mail	1.000	1.000	1.000	0.989	0.977	0.992
URL1	1.000	1.000	1.000	0.783	1.000	0.743
URL2	0.323	0.087	1.000	0.428	0.137	0.914
Price1	0.985	0.929	1.000	0.907	0.907	0.907
Price2	0.985	0.929	1.000	0.908	0.799	0.941

TABLE 7.7: The F2-score (F2), precision (P) and recall (R) for rule-based extraction of ICQ, jabber mail, URL and price attributes

ICQ:

The English and Russian extractors resulted in F2-scores of 1.000 and 0.973 respectively. The Russian extractor was unable to extract ICQ mentions that strongly deviated from the conventional writing style, e.g. by using Latin or Cyrillic characters to represent digits. Furthermore, most false positives were caused by year and price ranges in the posts, e.g. 2012-2014.

Jabber Mail:

The English and Russian extractors resulted in F2-scores of 1.000 and 0.989 respectively. It was found that BPH providers occasionally forget or intentionally omit white spaces or replace white spaces by special symbols, such as a dash. This resulted in boundary errors, since jabber mail addresses were extracted with excess characters. These excess characters that were mistaken as being part of the attribute were in fact legal characters for mail addresses. Therefore, the regular expression could not be updated to avoid these mistakes without causing other mistakes.

URL:

Two regular expressions were compared for URL extraction. These are 'URL1' and 'URL2'. The first regular expression resulted in the best performance for both English and Russian with a F2-score of 1.000 and 0.783 respectively. It was found that the Russian extractor resulted in several false negatives due to URLs that did not start with protocol information, e.g. 'http', followed by 'www'. The second regular expression was a copy of the first, but less restricted due to the fact that the protocol information and 'www' was made optional. However, even though the second regular expression increased the recall, it greatly worsened the precision and F2-score for both English and Russian.

Price:

Two regular expressions were compared for price extraction. These are 'Price1' and 'Price2'. The first regular expression scanned for numerical values followed of preceded by a currency. This resulted in a F2-score of 0.985 and 0.907 for English and Russian respectively. It was found that false negatives occurred due to a lack of currency or an unknown Russian currency in the sentence. Furthermore, when observing a price range, e.g. 100-200 USD, only the second price was extracted. False positives occurred due to ambiguous regular currency abbreviations, such as the Philippine peso (PHP) that matched PHP 5.

It was also found that prices are occasionally followed by a payment period. This was added in the second regular expression. The addition increased the recall of the Russian extractor with respect to the first regular expression, but worsened the precision due to new false positives. The F2-score of the Russian extractor only slightly increased using the second regular expression and there was no performance impact for the English extractor.

7.6 CRF Extractors

In this section, the results of the English and Russian CRF attribute extractors are presented and evaluated. As described in section 6.4, we started with an initial CRF extractor and conducted a series of experiments to improve the performance of the initial CRF extractor. These experiments explored the impact of the machine learning algorithm, sequence definition, hyper-parameter tuning, feature engineering and data pre-processing. Afterwards, the outcomes of all experiments were used to develop the final CRF extractor.

The following paragraphs show and evaluate the results of each experiment. Afterwards, the final CRF is described and its performance compared with that of the initial CRF extractor.

Experiment 1: Machine Learning Algorithms

In the first experiment, a sequence model was compared against a regular classifier, the initial CRF extractor and the MEM extractor respectively. For both English and Russian, it was found that the initial CRF extractor outperformed the MEM extractor with respect to the $F2_{Mi}$-score as shown in table 7.8. This indicates that knowing the predicted label of the preceding token is helpful.

	ModelInfo	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
En	CRF	0.697	0.829	0.670	0.387	0.541
	MEM	0.544	0.724	0.512	0.270	0.408
Ru	CRF	0.777	0.904	0.751	0.582	0.694
	MEM	0.612	0.729	0.589	0.448	0.547

TABLE 7.8: Performance scores of the initial CRF extractor and MEM extractor

Experiment 2: Sequence Definition

In the second experiment, the impact of using three different sequence definitions was explored. These are the post, paragraph and sentence sequence definitions. For both English and Russian, it was found that the sentence sequence definition resulted in the best performance with a $F2_{Mi}$-score of 0.778 and 0.832 respectively as shown in table 7.9. This indicates that knowing context from the previous and following sentences or paragraphs is not beneficial for attribute extraction.

	Sequence definition	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
En	Post	0.697	0.829	0.670	0.387	0.541
	Paragraph	0.726	0.862	0.698	0.389	0.565
	Sentence	0.778	0.896	0.753	0.535	0.633
Ru	Post	0.777	0.904	0.751	0.582	0.694
	Paragraph	0.807	0.903	0.786	0.606	0.722
	Sentence	0.832	0.917	0.813	0.671	0.752

TABLE 7.9: CRF extractor - The performance impact of post, paragraph and sentence sequence definitions

Experiment 3: Hyper-parameter Tuning

The third experiment explored the impact of hyper-parameter tuning using the sentence sequence definition. For both English and Russian, it was found that the tuned hyper-parameters resulted in a slight performance improvement compared to using the default hyper-parameter values of the initial CRF extractor. The results are shown in table 7.10. The hyper-parameters and their tuned values can be found in appendix A.

	Hyper-parameters	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
En	Default	0.778	0.896	0.753	0.535	0.633
	Tuned	0.781	0.902	0.755	0.538	0.638
Ru	Default	0.832	0.917	0.813	0.671	0.752
	Tuned	0.838	0.929	0.818	0.668	0.758

TABLE 7.10: CRF extractor - The performance impact of hyper-parameter tuning

Experiment 4: Feature Engineering

The fourth experiment tested the impact of extending the features in the feature vectors using the sentence sequence definition and tuned hyper-parameters. This process consisted of feature development to create new features and feature selection to select the features that are beneficial for the attribute extraction performance. The results are shown in table 7.11. The first row shows the performance when using the features of the initial CRF extractor as described in section 6.4.1. The second row shows the performance when using all features including context features of the four neighbouring tokens. The third row shows the performance of the CRF extractor with only the beneficial features, i.e. all features excluding the features without

a positive performance impact.

For both English and Russian, it was found that the CRF extractor using all features outperformed the CRF using the initial features. Furthermore, using beneficial features instead of all features resulted in a performance improvement for Russian. However, for English this was not the case. From this is can be inferred that the CRF was able to tackle the problem of unbeneficial features, e.g. by using possible beneficial feature dependencies that include these features.

	Feature set	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
En	Initial features	0.781	0.902	0.755	0.538	0.638
	All features	0.811	0.757	0.825	0.531	0.639
	Beneficial features	0.809	0.889	0.791	0.541	0.651
Ru	Initial features	0.838	0.929	0.818	0.668	0.758
	All features	0.857	0.760	0.885	0.707	0.765
	Beneficial features	0.861	0.915	0.849	0.712	0.777

TABLE 7.11: CRF extractor - The performance impact of using the initial, all or all beneficial features

The impact of adding context features from neighbouring tokens was also explored. The results are shown in table 7.12. For this we used all features for the English extractor and all beneficial features for the Russian extractor. The window sizes that were tested are 1, 3 and 5 tokens. For both English and Russian, it was found that adding context features improves the performance. However, for English a window of 3 resulted in the best $F2_{Mi}$-score and for Russian there was no difference in $F2_{Mi}$-score when using window 3 or 5.

	Context	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
En	window 1	0.788	0.723	0.806	0.548	0.641
	window 3	0.820	0.773	0.832	0.553	0.661
	window 5	0.811	0.757	0.825	0.531	0.639
Ru	window 1	0.848	0.904	0.835	0.699	0.766
	window 3	0.861	0.916	0.849	0.719	0.780
	window 5	0.861	0.915	0.849	0.712	0.777

TABLE 7.12: CRF extractor - The performance impact of using context features with a window of 1, 3 and 5

Experiment 5: Data Pre-processing

The fifth experiment explored the impact of five data pre-processing steps. These are lowercasing, stemming, BB-code removal, punctuation removal and stopword removal. The results are shown in table 7.13. For English, it was found that none of these pre-processing steps improved the $F2_{Mi}$-score. Instead, pre-processing removed characteristics from tokens that were beneficial for the English CRF extractor. For the Russian CRF extractor, stemming increased the $F2_{Mi}$-score from 0.861 to 0.870. The other pre-processing steps did not have an impact on the performance of worsened the performance.

	Pre-processing step	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
En	None	0.820	0.773	0.832	0.553	0.661
	Lowercasing	0.809	0.759	0.823	0.552	0.642
	Stemming	0.807	0.753	0.822	0.547	0.647
	BB-code removal	0.814	0.765	0.827	0.551	0.652
	Punctuation removal	0.801	0.754	0.813	0.551	0.650
	Stopword removal	0.792	0.771	0.798	0.546	0.629
Ru	None	0.861	0.913	0.848	0.717	0.779
	Lowercasing	0.861	0.913	0.848	0.712	0.778
	Stemming	0.870	0.921	0.858	0.719	0.785
	BB-code removal	0.859	0.913	0.847	0.710	0.778
	Punctuation removal	0.848	0.911	0.834	0.716	0.767
	Stopword removal	0.857	0.906	0.845	0.715	0.775

TABLE 7.13: CRF extractor - The performance impact of data pre-processing steps

Initial CRF Extractor vs. Final CRF Extractor

The final CRF extractor was developed by incorporating the results of the experiments. The English final CRF extractor uses the sentence sequence definition, tuned hyper-parameters as shown in appendix A, all features with a window of 3 and no data pre-processing. The Russian final CRF extractor uses the sentence sequence definition, tuned hyper-parameters, beneficial features with a window of 3 and stemming as a pre-processing step.

For both English and Russian, in was found that the final CRF extractor outperformed the initial CRF extractor with respect to the $F2_{Mi}$-score as shown in table 7.14. For Russian, the F2-score of each individual attribute was improved by the final CRF extractor. For English, most attributes improved. The F2-score of the attributes ICQ, payment method and jabber mail worsened and the F2-score of URL, nickname and service details remained unchanged.

7.7 Hybrid CRF Extractors

In this section, the results of the English and Russian hybrid CRF extractors are presented and evaluated. The hybrid CRF extractor was created by adding hybrid features to the final CRF extractor from the previous section in an attempt to improve the performance. As described in section 6.5, we experimented with adding dictionary, rule and transfer learning features. The outcomes of these experiments were used to develop the final hybrid CRF extractors for English and Russian. In the following paragraphs the impact of each individual dictionary and rule feature is evaluated followed by that of each individual transfer learning feature. Afterwards, the final hybrid CRF extractor is described and compared with the final CRF extractor from section 7.6.

Dictionary and Rule Features

Attribute	English		Russian	
	Initial	Final	Initial	Final
URL	0.000	0.000	0.726	0.729
ICQ	0.880	0.816	0.844	0.945
Nickname	0.000	0.000	0.366	0.684
CommunicationChannel	0.802	0.905	0.872	0.915
TechSpecification	0.486	0.712	0.705	0.807
Service	0.769	0.888	0.903	0.939
Content	0.698	0.787	0.673	0.754
Location	0.588	0.603	0.764	0.841
PaymentPeriod	0.714	0.882	0.645	0.774
Currency	0.200	0.794	0.864	0.941
PaymentMethod	0.636	0.625	0.137	0.400
JabberMail	1.000	0.898	0.936	0.974
ServiceDetails	0.000	0.000	0.201	0.677
OtherContactDetails	0.000	1.000	0.746	0.882
Support	0.321	0.577	0.357	0.787
Price	0.185	0.703	0.819	0.934
$F2_{Mi}$	0.697	0.820	0.777	0.870

TABLE 7.14: Final CRF extractor and initial CRF extractor F2-scores

It was found that for both English and Russian, one of the dictionary features resulted in a slightly improved $F2_{Mi}$-score as shown in table 7.15. These are the location and payment period dictionary features respectively. None of the rule features resulted in a performance improvement.

		AddedFeature	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
English		None	0.820	0.773	0.832	0.553	0.661
	Dict	Location	0.825	0.789	0.834	0.556	0.663
	Dict	Currency	0.819	0.780	0.829	0.554	0.658
	Dict	Period	0.818	0.770	0.831	0.557	0.660
	Rule	URL	0.820	0.775	0.832	0.556	0.660
	Rule	ICQ	0.814	0.773	0.825	0.546	0.648
	Rule	Mail	0.819	0.779	0.829	0.553	0.659
	Rule	Price	0.816	0.776	0.827	0.554	0.657
Russian		None	0.870	0.921	0.858	0.719	0.785
	Dict	Location	0.868	0.920	0.856	0.720	0.783
	Dict	Currency	0.870	0.918	0.859	0.721	0.786
	Dict	Period	0.871	0.921	0.859	0.722	0.786
	Rule	URL	0.870	0.919	0.858	0.718	0.784
	Rule	ICQ	0.867	0.916	0.855	0.719	0.782
	Rule	Mail	0.870	0.920	0.858	0.719	0.785
	Rule	Price	0.868	0.921	0.856	0.719	0.785

TABLE 7.15: Hybrid CRF extractor - The performance impact of dictionary and rule features

Transfer Learning Features

It was found that performing transfer learning by adding similar word features resulted in an improved $F2_{Mi}$-score. For English, this holds for the transfer learning features that add 1 up to 10 similar words to the feature vector. For Russian, this holds for the transfer learning features that add 10 up to 40 similar words. The other transfer learning features did not have a performance impact or worsened the performance. The results of all transfer learning features are shown in table 7.16.

	AddedFeature	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
English	None	0.820	0.773	0.832	0.553	0.661
	POS	0.815	0.774	0.826	0.552	0.653
	embedding	0.804	0.743	0.821	0.542	0.648
	1similar	0.823	0.789	0.832	0.560	0.663
	5similar	0.821	0.785	0.830	0.557	0.661
	10similar	0.824	0.791	0.833	0.559	0.665
	20similar	0.815	0.781	0.824	0.545	0.657
	40similar	0.820	0.790	0.827	0.543	0.658
Russian	None	0.870	0.921	0.858	0.719	0.785
	POS	0.868	0.915	0.857	0.719	0.783
	embedding	0.868	0.905	0.859	0.717	0.782
	1similar	0.870	0.918	0.859	0.723	0.786
	5similar	0.870	0.921	0.858	0.719	0.786
	10similar	0.871	0.918	0.860	0.717	0.785
	20similar	0.876	0.920	0.865	0.728	0.790
	40similar	0.877	0.921	0.866	0.736	0.791

TABLE 7.16: Hybrid CRF extractor - The performance impact of transfer learning features

Hybrid CRF Extractor vs. CRF Extractor

For English, the final hybrid CRF extractor was developed by adding the location dictionary feature to the English final CRF extractor. For Russian, the final hybrid CRF extractor was developed by adding the 40 similar words transfer learning feature to the Russian final CRF extractor. It was found that for both English and Russian, the final hybrid CRF extractors slightly outperformed the final CRF extractors from section 7.6 with respect to the $F2_{Mi}$-score as shown in table 7.17. Using the English hybrid CRF extractor, the F2-score improved for six attributes and decreased for three attributes. For the Russian hybrid CRF extractor, the F2-score improved for eleven attributes and decreased for three attributes.

Noticeable is that none of the other beneficial features from tables 7.15 and 7.16 are part of the final hybrid CRF extractors. This is due to the fact that adding the best transfer learning feature in combination with the best dictionary feature resulted in a slightly worsened performance for both English and Russian. An influencing factor is the finding that the performance impact of hybrid features is attribute dependent. For example, whereas the 40 similar words transfer learning feature improved the

F2-score of the Russian extractor for the location attribute, the period dictionary feature worsened this F2-score. Therefore, once combined, some improvements caused by one feature were neutralized by the other.

	English models		Russian models	
Attribute	CRF	Hybrid	CRF	Hybrid
URL	0.000	0.000	0.729	0.738
ICQ	0.816	0.816	0.945	0.953
Nickname	0.000	0.000	0.684	0.723
CommunicationChannel	0.905	0.930	0.915	0.907
TechSpecification	0.712	0.689	0.807	0.818
Service	0.888	0.890	0.939	0.932
Content	0.787	0.803	0.754	0.771
Location	0.603	0.638	0.841	0.877
PaymentPeriod	0.882	0.833	0.774	0.774
Currency	0.794	0.847	0.941	0.954
PaymentMethod	0.625	0.625	0.400	0.571
JabberMail	0.898	0.916	0.974	0.979
ServiceDetails	0.000	0.000	0.677	0.680
OtherContactDetails	1.000	1.000	0.882	0.882
Support	0.577	0.570	0.787	0.776
Price	0.703	0.703	0.934	0.944
$F2_{Mi}$	0.820	0.825	0.870	0.877

TABLE 7.17: Final CRF extractor and final hybrid CRF extractor F2-scores

7.8 biLSTM-CRF Extractors

In this section, the results of the English and Russian biLSTM-CRF extractors are presented and evaluated. As described in section 6.6, we started with an initial biLSTM-CRF extractor and conducted experiments to improve its performance. These experiments explored the impact of the sequence definition, hyper-parameter values, architecture and data pre-processing. Afterwards, the outcomes of all experiments were used to develop the final biLSTM-CRF extractor.

The following paragraphs show and evaluate the results of each experiment. Afterwards, the final biLSTM-CRF extractor is described and its performance compared with that of the initial biLSTM-CRF extractor.

Experiment 1: Sequence Definition

The first experiment explored the impact of using the post, paragraph and sentence sequence definitions. For both English and Russian, it was found that the sentence sequence definition resulted in the best performance with a $F2_{Mi}$-score of 0.728 and 0.822 respectively. The results are shown in table 7.18.

	Sequence	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
EN	Post	0.433	0.581	0.407	0.065	0.264
	Paragraph	0.613	0.734	0.588	0.201	0.455
	Sentence	0.728	0.764	0.719	0.445	0.584
RU	Post	0.732	0.815	0.714	0.542	0.651
	Paragraph	0.775	0.839	0.760	0.600	0.698
	Sentence	0.822	0.862	0.812	0.671	0.739

TABLE 7.18: biLSTM-CRF extractor - The performance impact of post, paragraph and sentence sequence definitions

Experiment 2: Hyper-parameters and Architecture

The second experiment explored with adapting the batch size, dropout, embedding dimension and epoch hyper-parameters. Furthermore, the impact of using a single LSTM instead of a bidirectional LSTM and adding character embeddings was tested. The experiments were conducted with the initial biLSTM-CRF extractor that makes use of a sentence sequence definition. The results are shown in table 7.19.

For English, using a batch size of 8 instead of 32 improved the $F2_{Mi}$-score as well as using both the regular and recurrent dropout, especially with a lowered dropout rate of 0.15. Increasing the word embedding dimension and adding character embeddings improved the $F2_{Mi}$-score as well. Furthermore, it was beneficial to increase the number of epochs and to use the bidirectional LSTM instead of a single LSTM.

For Russian, the findings are similar to those of the English biLSTM-CRF extractor. It was found that using a batch size of 8, lowering the dropout rate to 0.25, increasing the number of epochs, using a bidirectional LSTM and adding character embeddings was beneficial for the performance. However, increasing the embedding dimension did not have much impact on the performance.

Experiment 3: Data Pre-processing

The third experiment explored with the data pre-processing steps lowercasing, stemming, BB-code removal, punctuation removal and stopword removal. The experiments were conducted with the initial biLSTM-CRF extractor with a sentence sequence definition. The results are shown in table 7.20.

For both English and Russian, it was found that lowercasing and stemming improved the performance.

	Experiment	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
English	*Initial extractor*	0.728	0.764	0.719	0.445	0.584
	Initial with batch 32	0.691	0.754	0.676	0.268	0.528
	Initial with dropout .25	0.727	0.749	0.722	0.416	0.580
	Initial with dropout .15*	0.741	0.768	0.734	0.429	0.591
	Initial with no recurrent dropout	0.714	0.732	0.710	0.406	0.565
	Initial with no regular dropout	0.726	0.787	0.713	0.428	0.586
	Initial with embedding dim 200	0.734	0.775	0.724	0.442	0.587
	Initial with embedding dim 300*	0.738	0.780	0.729	0.479	0.596
	Initial with 100 epochs	0.753	0.768	0.750	0.462	0.601
	Initial with 200 epochs*	0.763	0.789	0.757	0.497	0.610
	Initial with 1 LSTM (no bidirectional)	0.718	0.663	0.733	0.405	0.560
	Initial with character embeddings*	0.764	0.772	0.762	0.489	0.603
Russian	*Initial extractor*	0.822	0.862	0.812	0.671	0.739
	Initial with batch 32	0.816	0.848	0.809	0.658	0.736
	Initial with dropout .25*	0.825	0.863	0.817	0.662	0.743
	Initial with dropout .15	0.820	0.873	0.808	0.670	0.740
	Initial with no recurrent dropout	0.813	0.859	0.803	0.666	0.732
	Initial with no regular dropout	0.823	0.887	0.809	0.677	0.741
	Initial with embedding dim 200	0.823	0.883	0.809	0.670	0.741
	Initial with embedding dim 300	0.822	0.883	0.808	0.667	0.743
	Initial with 100 epochs	0.828	0.878	0.817	0.674	0.745
	Initial with 200 epochs	0.831	0.865	0.823	0.678	0.746
	Initial with 1 LSTM (no bidirectional)	0.810	0.837	0.803	0.660	0.726
	Initial with character embeddings*	0.838	0.861	0.833	0.692	0.754

TABLE 7.19: biLSTM-CRF extractor - The performance impact of adapting hyper-parameter values and architecture

	Experiment	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
English	*Initial extractor*	0.728	0.764	0.719	0.445	0.584
	Initial with lowercasing*	0.758	0.782	0.752	0.475	0.607
	Initial with stemming*	0.731	0.760	0.724	0.414	0.585
	Initial with BB-code removal	0.722	0.711	0.725	0.462	0.580
	Initial with punctuation removal	0.708	0.739	0.701	0.452	0.564
	Initial with stopword removal	0.700	0.728	0.693	0.415	0.546
Russian	*Initial extractor*	0.822	0.862	0.812	0.671	0.739
	Initial with lowercasing*	0.844	0.887	0.834	0.688	0.760
	Initial with stemming*	0.853	0.889	0.844	0.689	0.766
	Initial with BB-code removal	0.820	0.881	0.806	0.662	0.738
	Initial with punctuation removal	0.808	0.864	0.795	0.654	0.729
	Initial with stopword removal	0.818	0.863	0.807	0.674	0.738

TABLE 7.20: biLSTM-CRF extractor - The performance impact of data pre-processing steps

Initial biLSTM-CRF Extractor vs. Final biLSTM-CRF Extractor

The final English and Russian biLSTM-CRF extractors were developed by incorporating the results of the experiments. The experiments indicated with a * in tables 7.19 and 7.20 were used for the final biLSTM-CRF extractors. Furthermore, the sentence sequence definition was used. For Russian, the number of epochs was set at 25 instead of 200 due to computational constraints. Furthermore, we did not incorporate the embedding dimension of 200 and we used both regular and recurrent dropout.

For both English and Russian, it was found that the final biLSTM-CRF extractor outperformed the initial biLSTM-CRF extractor as shown in table 7.21. For both languages, the F2-scores of most attributes improved when using the final instead of initial extractor. The exceptions with a reduced F2-score are technical specification for English and URL, other contact details and support for Russian.

	English models		**Russian models**	
Attribute	Initial	Final	Initial	Final
URL	0.000	0.000	0.720	0.711
ICQ	0.652	0.913	0.878	0.934
Nickname	0.000	0.000	0.496	0.613
CommunicationChannel	0.916	0.974	0.886	0.921
TechSpecification	0.625	0.617	0.747	0.813
Service	0.812	0.911	0.895	0.927
Content	0.767	0.881	0.703	0.797
Location	0.481	0.496	0.767	0.860
PaymentPeriod	0.385	0.714	0.786	0.890
Currency	0.424	0.917	0.963	0.963
PaymentMethod	0.463	0.467	0.208	0.337
JabberMail	0.911	0.964	0.968	0.983
ServiceDetails	0.000	0.000	0.642	0.691
OtherContactDetails	1.000	1.000	0.870	0.797
Support	0.563	0.570	0.710	0.697
Price	0.328	0.738	0.871	0.920
$F2_{Mi}$	0.728	0.804	0.822	0.872

TABLE 7.21: Initial biLSTM-CRF extractor and final biLSTM-CRF extractor F2-scores

7.9 Transfer Learning with the biLSTM-CRF Extractors

This section evaluates the impact of performing transfer learning, by fine-tuning, on the final biLSTM-CRF extractor from the previous section. As described in section 6.7, two experiments were conducted to test the applicability of transfer learning. The first experiment compared the impact of applying transfer learning on the word embedding layer versus both the word and character embedding layers. The second experiment tested the impact of using various source datasets for pre-training. The following paragraphs show and evaluate the results of the experiments. Afterwards, the results are compared with those of the final biLSTM-CRF extractor trained from

scratch.

Experiment 1: Transferring Word Embeddings vs. Word and Character Embeddings

For both English and Russian, it was found that transfer learning, with pre-training on the complete DNM as source dataset, worsened the performance. This is arguably due to the fact that the source dataset consists of mainly non-English posts and posts about unrelated topics. Of the two transfer learning attempts, transferring word embeddings resulted in a better performance compared to transferring both word and character embeddings.

	Transferred layer	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
EN	None	0.804	0.848	0.793	0.546	0.653
	Word only	0.779	0.849	0.764	0.522	0.631
	Word + character	0.767	0.837	0.751	0.499	0.626
RU	None	0.872	0.896	0.867	0.709	0.783
	Word only	0.866	0.891	0.860	0.689	0.760
	Word + character	0.864	0.886	0.859	0.696	0.770

TABLE 7.22: Transfer learning - The performance impact of transferring word embedding versus word and character embedding layers

Experiment 2: Comparing Source Datasets

The second source dataset consists of language filtered posts of the DNM, e.g. all English posts for English. For Russian, it was found that the second source dataset improved the performance with respect to the first source dataset. However, the $F2_{Mi}$-score was comparable to that of performing no transfer learning. For English, it was found that the second source dataset worsened the performance with respect to the first source dataset. This is arguably due to the posts about unrelated topics or due to the small size of the second source dataset.

The third source dataset consists of the language filtered and category filtered posts of the DNM, e.g. all English hosting posts for English. For Russian, the second and third source datasets resulted in a comparable $F2_{Mi}$-score. For English, it was found that the third source dataset improved the performance with respect to using the first or second source dataset. However, transfer learning still resulted in a worsened performance with respect to training from scratch.

The fourth source dataset consists of the language filtered and category filtered posts of multiple DNMs. For both Russian and English, it was found that the fourth source dataset improved the performance with respect to all other source datasets. Furthermore, for both languages and with the fourth source dataset, transfer learning resulted in a biLSTM-CRF extractor that outperformed the extractor when trained from scratch. This shows that transfer learning, by fine-tuning a transferred word embedding layer, can result in an improved performance compared to training from scratch if the source dataset consists of a sufficient amount of language filtered posts about a related topic.

	Source Dataset	$F2_{Mi}$	P_{Mi}	R_{Mi}	$F2_{Ma}$	$F2_{W}$
EN	All posts of 1 DNM	0.779	0.849	0.764	0.522	0.631
	English posts of 1 DNM	0.775	0.825	0.764	0.521	0.627
	English hosting posts of 1 DNM	0.790	0.835	0.780	0.540	0.647
	English hosting posts of >1 DNM	0.806	0.833	0.800	0.553	0.653
RU	All posts of 1 DNM	0.866	0.891	0.860	0.689	0.760
	Russian posts of 1 DNM	0.873	0.886	0.869	0.696	0.762
	Russian hosting posts of 1 DNM	0.872	0.890	0.868	0.698	0.747
	Russian hosting posts of >1 DNM	0.886	0.892	0.884	0.715	0.790

TABLE 7.23: Transfer learning - The performance impact of the source datasets

Chapter 8

Conclusion and Discussion

In this study, attribute extraction was performed on BPH posts of a DNM as a step towards effectively including DNMs in intelligence-led policing. For this purpose, a variety of attribute extraction techniques was tested and compared to find the best performing technique for each attribute. Furthermore, transfer learning was applied in an attempt to improve the attribute extraction performance.

Section 8.1 summarizes the results to answers the three research questions. Section 8.2 discusses the work including the limitations and improvements. Section 8.3 provides opportunities for future work.

8.1 Research Questions

RQ1: Which attributes are present in DNM posts advertising bulletproof hosting services?

English and Russian BPH posts contain both generic attributes, e.g. location, and BPH specific attributes, e.g. content. In total, sixteen attributes were identified, including twelve generic and four BPH specific attributes. This set consists of closed attributes, applicable for dictionary-based extraction, and open attributes. Open attributes can be extracted using either rule- and pattern-based extraction, in case of a distinctive structure, or machine learning based extraction.

RQ2: How well can these attributes be extracted from DNM posts advertising bulletproof hosting services?

In this study, we compared the performance of several extractors that used a variety of extraction techniques. All extractors were able to outperform the baseline extractor. The performance of the extractors is attribute dependent. Therefore, no extractor was able to extract all attributes and obtain their best F2-scores. For English, the ensemble extractor was able to outperform all other extractors with a $F2_{Mi}$-score of 0.859. For Russian, the biLSTM-CRF extractor developed with transfer learning resulted in the best performance with a $F2_{Mi}$-score of 0.886.

Most attributes were extracted with a F2-score greater than 0.850. For English, the exceptions are the attributes nickname(0.000), technical specification(0.714), payment method(0.636), service details(0.000) and support(0.625). For Russian, the exceptions are the attributes URL(0.783), nickname(0.723), technical specification(0.837), content(0.797), payment method(0.571), service details(0.693) and support(0.795).

RQ3: What is the impact of applying transfer learning on attribute extraction from DNM posts advertising bulletproof hosting services?

Transfer learning was applied to the CRF and the biLSTM-CRF extractors. For the CRF extractor, adding similar words based on word embeddings as features to the token feature vector resulted in a slight performance improvement. The $F2_{Mi}$-score improved from 0.820 to 0.824 for English and from 0.870 to 0.877 for Russian.

The biLSTM-CRF extractor developed with transfer learning by fine-tuning the word embedding layer was able to outperform the biLSTM-CRF extractor trained from scratch. This required a source dataset with language and category filtered posts from multiple DNMs to pre-train the word embedding layer. The $F2_{Mi}$-score slightly improved from 0.804 to 0.806 for English and improved from 0.872 to 0.886 for Russian.

8.2 Discussion and Limitations

For this study, we performed attribute extraction on 62 English and 470 Russian posts. Preferably, a larger dataset should be used. This is especially the case for the biLSTM-CRF extractors, as more data would have helped to improve the robustness of the extractors. Besides the dataset size, the class frequencies are also of importance. As was mentioned in section 5.5, there is a large class imbalence in both datasets. This is one of the reasons several attributes were extracted with a low F2-score and some attributes could not be extracted.

The data consists of posts from 27 threads created by a total of 23 BPH providers. Furthermore, whereas the majority of threads have five or fewer posts, some threads were very active with a post count up to 268. This caused a disproportionate representation of authors in the dataset. In case the 23 authors have a writing style that is distinct from other authors, this could result in bad generalizability to new authors. If the writing style of these new authors is in fact distinct, we can speak of having different domains.

We attempted to develop attribute extraction models for English and Russian separately. However, the automatic split of posts did not result in making the two datasets fully English or Russian. This is due to the fact that some authors interchanged both languages in their post. To keep the structure of the posts, this did not result in a split of the post. One can argue that part of the English domain is in this case used to train the Russian attribute extraction model and vice versa. This could result in a better cross-lingual applicability of the extractors, similar to the successful transfer learning experiment performed by Durrett et al. [10] described in section 3.3.2.

Due to the similarity in research performed by Portnoff et al.[9], the performance of the three price related attributes could be compared. The attributes are payment method, price(amount) and currency. Portnoff et al. extracted price information using regular expression and SVM based extraction. The regular expression extractor, developed to extract numbers and known currencies, resulted in a low F1-score varying between 0.296 and 0.356 depending on the darknet forum. The SVM extractor, trained to extract prices and payment methods including currencies, resulted in

F1-scores between 0.988 and 0.729. It was found that the regular expression extractor lacked context information required to distinguish between prices and regular numbers and to capture the various ways of presenting price information in posts. In our study, the prices were extracted, using regular expressions, with a F1-score of 0.963 for English. This performance was the result of including context information that the regular expression of Portnoff et al. lacked. More specifically, the numerical values should be preceded or followed by a currency. For Russian, prices were extracted with a F1-score of 0.937, using an ensemble extractor, compared to 0.907 with a regular expression.The result of our payment method extraction, 0.683 for English (CRF) and 0.615 for Russian (Hybrid), is far below the result Portnoff et al. achieved with their SVM model. The results of our currency extraction, 0.917 for English and 0.960 for Russian (biLSTM-CRF), is comparable to that of the best SVM model developed by Portnoff et al.

In this study, hyper-parameter optimization was performed in two phases, tuning discrete hyper-parameters and continuous hyper-parameters separately. However, ideally, hyper-parameter tuning is performed for both hyper-parameter types at the same time. Furthermore, a more extensive hyper-parameter optimization can be performed, since in this research we tested three up to five values per continuous hyper-parameter.

Feature selection tested one feature at a time and therefore did not take into account possible dependencies between features. An alternative is to perform an incremental feature selection. At each iteration the feature with the largest performance improvement is added. Without removing the feature for the next iteration, as we did due to time and computational constraints.

Another impact on the choice of extraction technique can be how long each technique takes to train. This was not included in our study. However, in case of limited time, this might be useful as training the dictionary-based and rule- and pattern-based extractors takes a few seconds, training the CRF takes a few hours for Russian and training the biLSTM-CRF with transfer learning takes a few days for Russian.

Furthermore, one might question the ethicality of using machine learning to gain insight in DNMs. For example, are we allowed to actively explore cross-cutting cybercrime enablers if the providers do not necessarily perform illegal activities themselves? Are we allowed to use neural networks for law enforcement operations, since they are often compared to a black box with no way to trace how a decision has been made by the network.

8.3 Future Work

In recent work by Howard et al. [33] a new method named Universal Language Model Fine-Tuning was developed. This method is focussed on performing transfer learning on neural network models trained to perform language processing tasks. This method was able to outperform state-of-the-art text classification tasks including topic classification on a DBpedia dataset and sentiment analysis on an IMDb dataset. Furthermore, it is argued to be equally applicable for sequence labelling tasks. This method is a great opportunity for future work. The method is introduced (for text classification) in the remainder of this paragraph.

The method consists of three parts: Language Model (LM) pre-training, LM fine-tuning, and classifier fine-tuning. Language modelling is the task of predicting token n+1, e.g. a word or character, given the preceding sequence of n tokens. The first phase trains a LSTM language model on a large, domain general dataset consisting of approximately 28.000 Wikipedia articles. The goal is to learn general characteristics of the language. The second phase transfers the pre-trained LSTM to the target domain through fine-tuning on the domain specific target data. In general, fine-tuning updates weights of the last layer, the last several layers or all layers of the LSTM. This results in a language model for the target domain that learned target specific characteristics. The last phase transfers the LSTM to perform the desired task, a classifier task in their case. In this phase two 'classifier layers' are added and their weights learned from scratch. Previously learned weights are fine-tuned using gradual unfreezing to prevent overfitting and catastrophic forgetting. Weights of frozen layers are not updated. With the gradual unfreezing method, layers are defrosted one layer per epoch ending at the initial layer that captures the more generic knowledge.

Another opportunity for future work is to perform single-attribute optimization of the extractors. For this study, we focussed on performing experiments to improve the $F2_{Mi}$-score of the extractors. However, this resulted in a disadvantage for attributes with a low class frequency and an advantage for attribtutes with a high class frequency. For example, we used the sentence sequence definition as that was found to outperform paragraph-level and post-level sequences. However, there might be attributes that benefit more from a paragraph-level sequence. For example, payment methods, that are more often represented in a listing, with one attribute per sentence. Instead, extractors can be optimized for each attribute separately and the results of this study can be used as a starting point for this.

A third opportunity, since the majority of our attributes is not BPH specific, is to apply our extractors in a cross-domain or cross-language setting to see how good our extractors generalize. Furthermore, transfer learning can be applied to perform domain adaptation as was attempted by Durrett et al. [10] and to make the extractors more domain independent.

Appendix A

Hyper-parameter Tuning

Discrete hyper-parameters tested:

Hyper-parameter	Values	Tuned value English	Tuned value Russian
Algorithm	lbfgs, l2sgd, ap, pa, arow	lbfgs	lbfgs
All possible states	True, False	True	True
All possible transitions	True, False	True	False
Linesearch*	MoreThuente, Backtracking, StrongBacktracking	MoreThuente	MoreThuente
error_sensitive**	True, False	-	-
Averaging**	True, False	-	-

TABLE A.1: Hyper-parameter tuning phase 1: Discrete Hyper-parameters with the tested parameter values and the tuned values

* lbfgs hyper-parameter
** pa hyper-parameter

Continuous hyper-parameters tested:
For this we used the tuned discrete hyper-parameter values shown in table A.1.

Hyper-parameter	Values	Tuned value English	Tuned value Russian
c1	[0.0, 0.05, 0.1, 0.4, 1.0]	0.05	0.05
c2	[0.0, 0.05, 0.1, 0.4, 1.0]	0.05	0.1
max_iterations	[50, 100, 150, 250]	150	250
max_linesearch	[10, 20, 40]	10	10
min_freq	[0, 5, 15]	0	0
num_memories	[3,6,9]	9	6
epsilon	[0.000001, 0.00001, 0.0001]	1e-06	1e-06
period	[5, 10, 20]	5	5
delta	[0.000001, 0.00001, 0.0001]	1e-06	1e-06

TABLE A.2: Hyper-parameter tuning phase 2: Continuous Hyper-parameters with the tested parameter values and the tuned values

Appendix B

Feature Engineering

nr	Feature	Type	Description
1	Token	w	Token value
2	IndexInSeq	d	Index of the token in the sequence
3	Prefix3char	w	First three characters of the token
4	Suffix3char	w	Last three characters of the token
5	ContainsCap	w	Does the token contain a capital letter?
6	ContainsDigit	w	Does the token contain a digit?
7	ContainsSpecialChar	w	Does the token contain a special character (non word character, non digit and non whitespace)
8	IsInStopwords	l	Is the token a Russian or English stopword
9	TokenFrequency	d	Number of occurrences of the token in the dataset

TABLE B.1: Initial feature set

nr	Feature	Type	Description
10	Lower	w	Lowercased token value
11	Stem	w	Stem of the token value
12	Pattern_Latin(l)Cyrillic(c)	w	Mapping characters to a small set of symbols representing capital and lower Latin(Ll) and Cyrillic letters(Cc), digits(d) and other characters (o): f(BBbБб-26)=LLlCcodd
13	Pattern_short_Latin(l)Cyrillic(c)	w	Summarized version of the Pattern_Latin(l)Cyrillic(c) feature, represent a sequence of two or more identical pattern symbols as one symbol followed by '+': f(BBbБб-26)=L+lCcod+
14	Pattern_LatinCyrillic(a)	w	Mapping characters to a small set of symbols representing capital letters (A), lower letters (a), digits(d) and other characters (o): f(BBbБб-26)=AAaAaodd
15	Pattern_short_LatinCyrillic(a)	w	Summarized version of the Pattern_LatinCyrillic(a) feature, represent a sequence of two or more identical pattern symbols as one symbol followed by '+': f(BBbБб-26)=A+aAaod+
16	Prefix1char	w	First character of the token
17	Prefix2char	w	First two characters of the token
18	Suffix1char	w	Last character of the token
19	Suffix2char	w	Last two characters of the token
20	ContainsLatin	w	Does the token contain a letter in Latin script?
21	ContainsCyrillic	w	Does the token contain a letter in Cyrillic script?
22	IsFirstCap	w	Is the first character a capital letter?
23	IsLastCap	w	Is the last character a capital letter?
24	IsAllLower	w	Are all letters lowercased?
25	IsAllCap	w	Are all letters capitalized?
26	IsLowerAndCap	w	Does the token contain a capitalized and lowercased letter?
27	IsFirstDigit	w	Is the first character a digit?
28	IsLastDigit	w	Is the last character a digit?
29	IsAlphaAndDigit	w	Does the token contain a letter and a digit?
30	IsAllDigit	w	Are all characters digits?
31	ContainsDot	w	Does the token contain a dot?
32	ContainsAtSign	w	Does the token contain an at sign (@)?

TABLE B.2: Extended feature set part 1

nr	Feature	Type	Description
33	ContainsDollarSign	w	Does the token contain a dollar sign?
34	ContainsParenthesis	w	Does the token contain parenthesis? - (or)
35	ContainsHyphen	w	Does the token contain a hyphen?
36	ContainsSquareBrackets	w	Does the token contain square brackets? - [or]
37	ContainsForwardSlash	w	Does the token contain a forward slash?
38	CharBiGrams	w	A feature for each unique character bi-gram in the token f(token)=[to,ok,ke,en]
39	CharTriGrams	w	A feature for each unique character tri-gram in the token f(token)=[tok,oke,ken]
40	UniqueCharacters	w	A feature for each unique character in the token f(tokentoken)=[t,o,k,e,n]
41	TokenFrequency_lower	d	Number of occurrences of the lowercased token in the dataset
42	SequenceLength	d	The number of tokens in the sequence
43	PositionOfTokenInSequence_Percentage	d	(token index in the sequence+1)/number of tokens in the sequence
44	PositionOfTokenInSequence_Quartiles	d	Is the token in the first, second, third or fourth quarter of the sequence?
45	PositionOfTokenInSequence_Halves	d	Is the token in the first or second half of the sequence?
46	IsInListing	d	Does the sequence start with: -, •, digit., digit), +, * or [*]?
47	TokensRemovedFromPreviousPunctuation	d	How many tokens removed from the previous punctuation in the sequence?
48	TokensUntilNextPunctuation	d	How many tokens until the next punctuation in the sequence?
49	IsInKnownValues_Content	l	Is the token a known content attribute
50	IsInKnownValues_CommunicationChannel	l	Is the token a known communication channel attribute
51	IsInKnownValues_Currency	l	Is the token a known currency attribute
52	IsInKnownValues_ICQ	l	Is the token a known ICQ attribute
53	IsInKnownValues_JabberMail	l	Is the token a known jabber mail attribute
54	IsInKnownValues_Location	l	Is the token a known location attribute
55	IsInKnownValues_Nickname	l	Is the token a known nickname attribute
56	IsInKnownValues_O	l	Is the token a known non-attribute
57	IsInKnownValues_OtherContactDetails	l	Is the token a known other contact details attribute
58	IsInKnownValues_PaymentMethod	l	Is the token a known payment method attribute
59	IsInKnownValues_PaymentPeriod	l	Is the token a known payment period attribute

TABLE B.3: Extended feature set part 2

nr	Feature	Type	Description
60	IsInKnownValues_Price	1	Is the token a known price attribute
61	IsInKnownValues_Service	1	Is the token a known service attribute
62	IsInKnownValues_ServiceDetails	1	Is the token a known service detail attribute
63	IsInKnownValues_Support	1	Is the token a known support attribute
64	IsInKnownValues_TechSpecification	1	Is the token a known technical specification attribute
65	IsInKnownValues_URL	1	Is the token a known URL attribute
66	CountInKnownValues_Content	1	How often is the token a content attribute
67	CountInKnownValues_CommunicationChannel	1	How often is the token a communication channel attribute
68	CountInKnownValues_Currency	1	How often is the token a currency attribute
69	CountInKnownValues_ICQ	1	How often is the token an ICQ attribute
70	CountInKnownValues_JabberMail	1	How often is the token a jabber mail attribute
71	CountInKnownValues_Location	1	How often is the token a content attribute
72	CountInKnownValues_Nickname	1	How often is the token a nickname attribute
73	CountInKnownValues_O	1	How often is the token a content attribute
74	CountInKnownValues_OtherContactDetails	1	How often is the token an other contact details attribute
75	CountInKnownValues_PaymentMethod	1	How often is the token a payment method attribute
76	CountInKnownValues_PaymentPeriod	1	How often is the token a payment period attribute
77	CountInKnownValues_Price	1	How often is the token a price attribute
78	CountInKnownValues_Service	1	How often is the token a service attribute
79	CountInKnownValues_ServiceDetails	1	How often is the token a service details attribute
80	CountInKnownValues_Support	1	How often is the token a support attribute
81	CountInKnownValues_TechSpecification	1	How often is the token a technical specification attribute
82	CountInKnownValues_URL	1	How often is the token an URL attribute

TABLE B.4: Extended feature set part 3

Appendix C

Annotation Guidelines

Annotation Guidelines: Bulletproof Hosting Attributes

This document contains the annotation guidelines in section 1 followed by annotation examples of each attribute in section 2.

Example annotated post: See ExamplePost.png

1. Annotation Guidelines:

General annotation guidelines:

- Only annotate *specific information* provided by the author, such as 'fast-flux' and 'hosting'. Vague terms such as 'other services' and 'tools' should be omitted.
- Annotate each attribute individually:
    - '**servers** and **domains'**
    - **VDS/VPS**
- Keep annotations as short as possible, unless removing preceding/following words removes information
    - **5 to 50GB**: annotating 5 on its own causes drastic loss of information
    - **256mb to 1gb**: no loss of information
    - 1GB – connection speed: removing connection speed causes drastic loss of information
- **Adjectives**: only include adjectives to the annotation if it provides new and relevant information about the attribute. For example: include bp (:bulletproof) in bp hosting. Omit adjectives that provide information that could already (implicitly) be inferred from the main attribute and that are buzz words. For example: do NOT include reliable in reliable hosting (or stable, comfortable etc.)

- **If an adjective refers to two attributes**: e.g. 'bulletproof hosting and domains'. Include the adjective to the first attribute only and keep the attributes split. Result: 'bulletproof hosting', 'domains'.

- **Two adjectives for one attribute (with at least one relevant adjective)**: e.g. 'dedicated and virtual servers'. Annotate the complete set of words.

- **Quotes from other users:** if it contains relevant properties, annotate them too.

- Do not annotate 'irrelevant' parts:
    - Quick installation – quick server setup
- Avoid including: ( )

2. Attributes

Service

Service are *all* mentions of offered services/products (explicit and implicit).

- Explicit: 'our services are bp hosting and dedicated servers'.
- Implicit: 'new servers in Malaysia' and 'choose from the following domains'.

Examples:

- Are sold SERVERS
- Available SERVERS in Malaysia
- Bp domain registration
- Bp hosting service
- arrangement of servers: Malaysia, jamaica… (12163_1 ??)
- Servers for virtual hosting are equipped with ..
- At the moment is accessible dedicated servers in Hongkong, Russia…
- The opportunity to gather server for your project
- Abuseproof dedicated servers
- VPN-service
- Autonomous systems (AS)
- abuse-servers

TechSpecification

Technical specifications are all server specifications (operating system, hdd, memory, speed, ip, etc.), control panel information and *techniques* mentioned.

Examples:

- P4 2.4 GHZ / 512 RAM / 1000 gb traffic / 1gb connection speed /5 IP + cPanel / WHM
- Core2due 2.1 1 gb RAM 50 gb HDD + panel
- CentOS 4.4 (Linux) (or any other OS)
- OS: linux
- OS: linux, *BSD, Windows
- Operating system: linux, *BSD, Windows
- - channel: 100mb/s
- 1GB – connection speed
- Connection speed 1GB (end of sentence)
- Control panel: directadmin – free
- Directadmin included
- Control panel: directadmin, pleks
- With Control panel…
- 3GB – RAM
- *sentence*: 'high speed of channels from 100mb to 1GB'.
- *Sentence*: HDD from **5 to 50GB**, memory from **256mb to 1gb**
- 100mbps speed
- RAID
- IRON
- Channels from SCNET and LEVEL3

- Server with 80GB HDD and limited domains*(=service)* and databases
- VPN
- IP blocking
- Anit-DDoS Software
- Dedicated IP
- Its Ip network

Do not annotate:

- OS – any
- Domains: 3 *(Domains=servic*e)

ServiceDetail

Service detail is information that distinguishes the offered service from others, excluding service and technical specification attributes.

Example:

- Daily backup of the data ..
- Backups every 24hours …
- Backups simply daily…
- Installation of additional hard drive
- Uptime 99%
- Partner and reseller programme
- We are no resellers
- Servers in our own datacenter
- (3 different data centers with different IP networks*(techspec)*)
- (private data center …)
- .com
- Load balancers and computer clusters
- Distributed network architecture

Do NOT annotate:

- We offer bulletproof and normal: hosting, servers, domains

Nickname

Nicknames are names given to the online presence of the seller or others.

PaymentMethod

Payment methods that can be used to pay for the service. Such as paypal and moneybookers.

Price

Price attributes convey information about the price of the service.
Example:

- 204$ B mecrl
- 120 to 200 dollar

Currency

Currency attributes are the currencies used to express the price of the service.
Example: 204$ B mecrl

PaymentPeriod

Payment period attributes show for which period a service is paid. Such as month or week.
Example: 204$ B mecrl

URL

URL attributes are URL mentions in posts to, e.g., link to the seller's webpage.
Examples:

- [URL]https://NAME.NL/shop[/URL]
- Ohtps:/name-name.nl

CommunicationChannel

Communication Channels are the online channels that can be used to contact the seller
Examples are ICQ and Jabber mail

ICQ

ICQ attributes are the ICQ accounts mentioned in the post
Examples:

- 000-000-000
- 000000

JabberMail

Jabber mail attributes are the jabber mail addresses mentioned in the post
Examples:

- name@name.info
- Name_name-name@name1.com

OtherContactDetails

Other contact details attributes are the other contact details provided in the post. Such as Skype contact details.
Examples:

- Name123
- Name-Name

Support

Support attributes provide information about the support offered by the seller:
Examples:

- Technical
- 24/7

Location

Location attributes contains all locations mentioned in the post. Including: cities, countries, Continents

Examples:

- Netherlands
- Amsterdam
- Chinese partners

Content

Content attributes describe the types of content that can and cannot be hosted. These contain specific content instances such as spam and exclude generic information such as ‘white projects’.

Examples:

- Any forms of spam
- For spam resources.
- SPAM using direct access
- Zooph
- projects directed towards breaking government organizations and organs of the executive power

Do not annotate: Any project

Appendix D

Baseline Results

	No overlap removed						Overlap removed					
	No lower (raw baseline)			Lower			No lower			Lower		
Attribute	F2	P	R	F2	P	R	F2	P	R	F2	P	R
Aggregated	0.502	0.487	0.519	0.472	0.420	0.539	0.651	0.739	0.583	0.656	0.698	0.619
overallscore	0.485	0.468	0.502	0.447	0.394	0.517	0.618	0.699	0.556	0.612	0.641	0.585
Service	0.485	0.394	0.641	0.438	0.340	0.624	0.811	0.832	0.791	0.832	0.838	0.827
ServiceDetails	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TechSpecification	0.240	0.549	0.157	0.190	0.191	0.194	0.240	0.549	0.157	0.201	0.215	0.194
Location	0.729	0.867	0.675	0.538	0.520	0.675	0.729	0.867	0.675	0.729	0.867	0.675
Price	0.150	0.117	0.217	0.150	0.117	0.217	0.150	0.117	0.217	0.150	0.117	0.217
Currency	0.267	0.300	0.250	0.467	0.500	0.450	0.267	0.300	0.250	0.467	0.500	0.450
PaymentPeriod	0.250	0.217	0.350	0.183	0.168	0.350	0.250	0.217	0.350	0.217	0.190	0.350
PaymentMethod	0.290	0.400	0.238	0.290	0.400	0.238	0.290	0.400	0.238	0.290	0.400	0.238
CommChannel	0.759	0.739	0.791	0.748	0.714	0.791	0.777	0.768	0.791	0.766	0.743	0.791
ICQ	0.533	0.600	0.500	0.533	0.600	0.500	0.533	0.600	0.500	0.533	0.600	0.500
Nickname	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
JabberMail	0.493	0.800	0.369	0.493	0.800	0.369	0.493	0.800	0.369	0.493	0.800	0.369
OtherContactDetails	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400
Content	0.818	0.921	0.746	0.828	0.874	0.795	0.818	0.921	0.746	0.848	0.917	0.795
Support	0.270	0.313	0.263	0.382	0.392	0.420	0.460	0.600	0.387	0.467	0.567	0.420
URL	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE D.1: Baseline performance scores - English

		No overlap removed						Overlap removed					
		No lower (raw baseline)			Lower			No lower			Lower		
	Attribute	F2	P	R	F2	P	R	F2	P	R	F2	P	R
0	Aggregated	0.485	0.395	0.629	0.477	0.368	0.681	0.705	0.693	0.719	0.719	0.679	0.764
1	overallscore	0.469	0.373	0.634	0.447	0.336	0.668	0.634	0.601	0.672	0.637	0.577	0.713
2	Service	0.554	0.424	0.804	0.533	0.397	0.815	0.860	0.836	0.888	0.862	0.822	0.910
3	ServiceDetails	0.216	0.174	0.362	0.199	0.141	0.393	0.371	0.388	0.369	0.369	0.356	0.405
4	TechSpecification	0.457	0.494	0.430	0.408	0.339	0.517	0.530	0.651	0.451	0.579	0.610	0.553
5	Location	0.800	0.903	0.721	0.685	0.629	0.776	0.818	0.943	0.726	0.779	0.787	0.787
6	Price	0.091	0.050	0.573	0.091	0.050	0.573	0.245	0.153	0.714	0.248	0.155	0.714
7	Currency	0.316	0.357	0.293	0.277	0.281	0.293	0.322	0.371	0.293	0.319	0.365	0.293
8	PaymentPeriod	0.237	0.150	0.715	0.224	0.141	0.715	0.306	0.213	0.735	0.291	0.202	0.735
9	PaymentMethod	0.480	0.800	0.385	0.433	0.667	0.435	0.480	0.800	0.385	0.433	0.667	0.435
10	CommChannel	0.655	0.530	0.884	0.621	0.470	0.914	0.714	0.607	0.892	0.672	0.529	0.922
11	ICQ	0.819	0.948	0.731	0.819	0.948	0.731	0.819	0.948	0.731	0.819	0.948	0.731
12	Nickname	0.596	0.768	0.526	0.550	0.595	0.546	0.645	0.866	0.546	0.583	0.674	0.546
13	JabberMail	0.850	0.978	0.771	0.850	0.978	0.771	0.850	0.978	0.771	0.850	0.978	0.771
14	OtherContactDetails	0.622	0.567	0.733	0.605	0.548	0.733	0.622	0.567	0.733	0.605	0.548	0.733
15	Content	0.416	0.338	0.603	0.425	0.332	0.640	0.509	0.455	0.657	0.530	0.456	0.703
16	Support	0.653	0.697	0.627	0.682	0.703	0.667	0.663	0.719	0.627	0.693	0.725	0.667
17	URL	0.133	0.200	0.100	0.133	0.200	0.100	0.147	0.300	0.100	0.147	0.300	0.100

TABLE D.2: Baseline performance scores - Russian

Bibliography

[1] J. Lewis, "Economic impact of cybercrime-no slowing down", *Santa Clara: McAfee & CSI (Center for Strategic and International Studies)*, 2018.

[2] A. K. Sood and R. J. Enbody, "Crimeware-as-a-service—a survey of commoditized crimeware in the underground market", *International Journal of Critical Infrastructure Protection*, vol. 6, no. 1, pp. 28–38, 2013.

[3] Europol, *Internet Organised Crime Threat Assessment 2018*. Europol, 2018.

[4] R. van Wegberg and T. Verburgh, "Lost in the dream? measuring the effects of operation bayonet on vendors migrating to dream market", in *Proceedings of the Evolution of the Darknet Workshop*, 2018, pp. 1–5.

[5] M. Graczyk and K. Kinningham, "Automatic product categorization for anonymous marketplaces", 2015.

[6] M. Spitters, F. Klaver, G. Koot, and M. van Staalduinen, "Authorship analysis on dark marketplace forums", in *Intelligence and Security Informatics Conference (EISIC), 2015 European*, IEEE, 2015, pp. 1–8.

[7] J. M. Giorgi and G. D. Bader, "Transfer learning for biomedical named entity recognition with neural networks", *Bioinformatics*, 2018.

[8] A. More, "Attribute extraction from product titles in ecommerce", *arXiv preprint arXiv:1608.04670*, 2016.

[9] R. S. Portnoff, S. Afroz, G. Durrett, J. K. Kummerfeld, T. Berg-Kirkpatrick, D. McCoy, K. Levchenko, and V. Paxson, "Tools for automated analysis of cybercriminal markets", in *Proceedings of the 26th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2017, pp. 657–666.

[10] G. Durrett, J. K. Kummerfeld, T. Berg-Kirkpatrick, R. S. Portnoff, S. Afroz, D. McCoy, K. Levchenko, and V. Paxson, "Identifying products in online cybercrime marketplaces: A dataset for fine-grained domain adaptation", *arXiv preprint arXiv:1708.09609*, 2017.

[11] D. Wang and T. F. Zheng, "Transfer learning for speech and language processing", in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*, IEEE, 2015, pp. 1225–1237.

[12] A. Greenberg, *Hacker lexicon: What is the dark web?*, Accessed: 2018-06-10, 2014. [Online]. Available: https://www.wired.com/2014/11/hacker-lexicon-whats-dark-web/.

[13] M. Chertoff and T. Simon, "The impact of the dark web on internet governance and cyber security", 2015.

[14] P. Lyman, "Archiving the world wide web", *Building a national strategy for digital preservation: Issues in digital media archiving*, pp. 38–51, 2002.

[15] G. Weimann, "Going dark: Terrorism on the dark web", *Studies in Conflict & Terrorism*, vol. 39, no. 3, pp. 195–206, 2016.

[16] T. J. Holt, O. Smirnova, and A. Hutchings, "Examining signals of trust in criminal markets online", *Journal of Cybersecurity*, vol. 2, no. 2, pp. 137–145, 2016. DOI: 10.1093/cybsec/tyw007. eprint: /oup/backfile/content_public/journal/cybersecurity/2/2/10.1093_cybsec_tyw007/2/tyw007.pdf. [Online]. Available: http://dx.doi.org/10.1093/cybsec/tyw007.

[17] V. Ciancaglini, M. Balduzzi, R. McArdle, and M. Rösler, "The deep web", *Trend Micro*, 2015.

[18] J. Jiang, "Information extraction from text", in *Mining text data*, Springer, 2012, pp. 11–41.

[19] S. Sarawagi *et al.*, "Information extraction", *Foundations and Trends® in Databases*, vol. 1, no. 3, pp. 261–377, 2008.

[20] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition", in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, Association for Computational Linguistics, 2009, pp. 147–155.

[21] S. Soderland, "Learning information extraction rules for semi-structured and free text", *Machine learning*, vol. 34, no. 1-3, pp. 233–272, 1999.

[22] Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Jagadish, "Regular expression learning for information extraction", in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2008, pp. 21–30.

[23] R. J. Mooney and R. Bunescu, "Mining knowledge from text using information extraction", *ACM SIGKDD explorations newsletter*, vol. 7, no. 1, pp. 3–10, 2005.

[24] D. Jurafsky and J. H. Martin, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall, 2009.

[25] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification", *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[26] C. N. d. Santos and V. Guimaraes, "Boosting named entity recognition with neural character embeddings", *arXiv preprint arXiv:1505.05008*, 2015.

[27] B. Tang, H. Cao, X. Wang, Q. Chen, and H. Xu, "Evaluating word representation features in biomedical named entity recognition tasks", *BioMed research international*, vol. 2014, 2014.

[28] E. F. Tjong Kim Sang and F. 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*, Association for Computational Linguistics, 2003, pp. 142–147.

[29] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition", *arXiv preprint arXiv:1603.01360*, 2016.

[30] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li, "Opentag: Open attribute value extraction from product profiles", *arXiv preprint arXiv:1806.01264*, 2018.

[31] C. C. Aggarwal and C. Zhai, *Mining text data*. Springer Science & Business Media, 2012.

[32] W Cohen and A. McCallum, "Information extraction from the world wide web", in *Tutorial Note of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, 2003.

[33] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification", *arXiv preprint arXiv:1801.06146*, 2018.

[34] J. Y. Lee, F. Dernoncourt, and P. Szolovits, "Transfer learning for named-entity recognition with neural networks", *arXiv preprint arXiv:1705.06273*, 2017.

[35] R. Nallapati, M. Surdeanu, and C. Manning, "Blind domain transfer for named entity recognition using generative latent topic models", in *Proceedings of the NIPS 2010 Workshop on Transfer Learning Via Rich Generative Models*, 2010, pp. 281–289.

[36] B. Tang, H. Cao, X. Wang, Q. Chen, and H. Xu, "Evaluating word representation features in biomedical named entity recognition tasks", *BioMed research international*, vol. 2014, 2014.

[37] T. Semwal, P. Yenigalla, G. Mathur, and S. B. Nair, "A practitioners' guide to transfer learning for text classification using convolutional neural networks", in *Proceedings of the 2018 SIAM International Conference on Data Mining*, SIAM, 2018, pp. 513–521.

[38] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A maximum entropy approach to natural language processing", *Computational linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[39] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", 2001.

[40] C. Sutton, A. McCallum, *et al.*, "An introduction to conditional random fields", *Foundations and Trends® in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012.

[41] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks", *arXiv preprint arXiv:1707.06799*, 2017.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[44] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging", *arXiv preprint arXiv:1508.01991*, 2015.

[45] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[46] N. Chinchor and B. Sundheim, "Muc-5 evaluation metrics", in *Proceedings of the 5th conference on Message understanding*, Association for Computational Linguistics, 1993, pp. 69–78.

[47] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, 1st ed. Cambridge University Press, 2008, ISBN: 0521865719.