Automatic Information Extraction and Relevance Evaluation of Epidemiological Texts Using Natural Language Processing

Auss Abbood

University Osnabrück,
Robert Koch Institute

This thesis is submitted in Mai 2019 for the degree of Master of Science

Abstract

To ensure permanent responsiveness to emerging disease outbreaks, the Robert Koch Institute (RKI) is the contact and evaluation point for notifications of reportable diseases from health departments and other public health institutes. However, the RKI also needs to look for emerging outbreaks in the vast amount of international epidemiological news to detect relevant outbreaks as early as possible. For this, a dedicated group of epidemiologists reads the news from several foreign sources every day, summarizes the most important articles, and then writes reports about them. The source and the key information about these articles like the described disease, country of origin, or confirmed cases are put into a curated list named Incident Database. This time-consuming procedure can be shortened by automated information extraction and assessment of epidemiological news. To this end, I have developed a surveillance tool which summarizes epidemiological news and rates the relevance of articles for the RKI.

Before I could develop the surveillance tool, I needed to acquire data to train machine learning models. Therefore, I used the annotations in the Incident Database and additionally scraped all articles from the most used sources of the Incident Database to create a labeled data set where articles put into the Incident Database are labeled *relevant*, and all the other scraped epidemiological articles are labeled *irrelevant*.

To summarize articles, I applied named entity recognition to epidemiological texts and then trained naive Bayes classifier using the Incident Database as expert labels to find the most important entities of an epidemiological article such as disease, country, and confirmed cases. For the relevance scoring, I compared naive Bayes classifiers using the bag-of-words approach with, a support vector machine, neural networks and other classifier trained with document embeddings of scraped and labeled articles. Finally, I built a web app to demonstrate the usability of this tool. With this, I showed how epidemiologists in the future could overcome the difficulty to process large amounts of international epidemiological intelligence.

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Osnabrück or any other University or similar institution except as specified in the text. I further state that no substantial part of my thesis has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Osnabrück or any other University or similar institution except as specified in the text.

Auss Abbood
April, 2019

Acknowledgements

In the beginning, I want to thank all the poor souls that needed to proofread my thesis or parts of it, or helped me in some other way finishing this thesis. Thank you, Knut, Rüdiger, Hannah, and in particular Alex, Stéphane, Patrick, and Jonas.

I thank everyone in the Signale team for making my time in Berlin so enjoyable. Thank you, Max, for welcoming me so openly, yet so eccentric. Thank you Fabse, for giving me a voice when I couldn't and understanding my idiosyncrasies. Thank you Stéphane and Alex for your excellent supervision. I really appreciated the freedom you gave me but also valued every time you took countermeasures. But mostly I valued that we met each other on eye level. I always felt respected by you and the rest of the RKI people.

My special thanks go to Rüdiger Busche with whom I shared my first student office. I enjoyed each of our conversations, your feedback, and all the other fun things we did in Berlin. You helped me grow and I really hope that our paths will cross again.

Contents

Glossary

ADASYN Adaptive Synthetic Sampling Approach for Imbalanced Learning.

CNN convolutional neural network.

CSS Cascading Style Sheets.

EBS event-based surveillance.

EDB Ereignisdatenbank
- *Incident Database*.

EpiLag Epidemiologische Bund-Länder-Lagekonferenz
- *Epidemiological Federal State-State Conference*.

EPIS Epidemic Intelligence Information System.

EU European Union.

GRITS Global Rapid Identification Tool System.

HTML Hypertext Markup Language.

IBA Index of Balanced Accuracy.

IfSG Infektionsschutzgesetz
- *Protection Against Infection Act*.

INIG Informationsstelle für Internationalen Gesundheitsschutz
- *The Information Centre for International Health Protection*.

IR information retrieval.

ML machine learning.

MLP multilayer perceptron.

NBC naive Bayes classifier.

NER named-entity recognition.

NLP natural language processing.

NLTK Natural Language Tool Kit.

POS-tagging position of speech tagging.

RASFF Rapid Alert System for Food and Feed.

regex regular expression.

REST Representational State Transfer.

RKI Robert Koch Institute.

SVM support vector machine.

tf-idf term frequency-inverse document frequency.

CHAPTER 1

Introduction

1.1 Epidemiological Surveillance at the Robert Koch Institute

The Robert Koch Institute (RKI), as a public health institute, is a central figure in the containment of health risks to the populace by being the main institution for disease prevention and surveillance in Germany. Its primary purposes are the detection, prevention, and control of infectious diseases [Robert Koch Institute, 2018]. To fulfill this objective, the priority tasks of the RKI lie in the field of epidemiology [Robert Koch Institute, 2018].

Epidemiology consists of a large field of studies that focuses on the distribution and the determinants of health-related events [WHO, 2014b]. Among many foci of epidemiology like health and demography, mental health, and non-communicable diseases, the primary interest during the proceedings of this thesis lies in **infectious epidemiology**. Its overall goal is the detection and (subsequent) containment of infectious disease outbreaks to minimize health consequences and the burden to the public health apparatus. The procedure to detect such potential risks could be expressed in simple terms as the detection of *occurrences of cases more than usual given time and region*. For the investigation and prevention of outbreaks, epidemiology makes use of several methodologies like descriptive studies and surveillance [WHO, 2014b].

One primary mission of infectious epidemiology is the early detection objective [WHO, 2014a]. Therefore, surveillance is an indispensable tool for a functional early warning mechanism. Since 2001, the **Infektionsschutzgesetz** (*the Protection against Infection Act*) (IfSG) is the foundation of the German surveillance system which demands a report to the authorities after the determination of a notifiable disease [Bundesminesterium der Justiz und für Verbraucherschutz, 2001]. The IfSG earmarked an electronic reporting system which led to the development of **SurvNet@RKI**, a web service and software tool

for reporting communicable diseases that is used by the local and state health departments of Germany [Faensen et al., 2006]. Finally, epidemiological surveillance is conducted by using the data accumulated by SurvNet@RKI to apply analyzes to detect conspicuities and disease outbreaks.

The provision of an infrastructure for epidemiologists to systematically detect, verify, and share data to acquire an informative picture of the epidemic threat to public health is called **epidemic intelligence** [WHO, 2014a]. It facilitates epidemiological surveillance through information processing and supply of formal, informal, actively and passively acquired information.

1.1.1 Indicator- and Event-Based Surveillance

The IfSG describes a traditional reporting system that facilitates the acquisition of trustworthy, human and non-human related health-based formal sources for the subsequent interpretation of such structured data [WHO, 2014a]. This process is called **indicator-based surveillance**. The acquisition of this data is mostly a passive process and follows routines established by the legislator and the public health institute which in the case of Germany is SurvNet@RKI. These routines follow rules that are disease- or syndrome-specific. Indicator-based surveillance is not only responsible for event detection but also for measuring the impact and evaluation of health programs [WHO, 2014a].

Hints for an outbreak can be detected through an increased amount of reported cases of a dangerous infection or changed circumstances that are known to entail disease outbreaks, e.g., increased reporting of salmonellosis during warm weather or (in the international context) a loss of proper sanitation which often leads to a cholera outbreak. Therefore, besides traditional surveillance that processes laboratory confirmations, external factors like weather, attendance monitoring at school and workplace, social media, and the web are also informative [WHO, 2014a].

The monitoring of information also generated outside the public health system, and its analysis is called **event-based surveillance**(EBS) and is the speed determining factor in epidemiological surveillance. With EBS, epidemiologists can detect and report events before the recognition of human cases in the reporting system of the public health system [WHO, 2014a]. The fast detection and verification of possibly threatening events are essential and heavily depend on a good-working epidemic intelligence to handle a large amount of data from various sources. Especially in the times of the internet, the topicality and quantity of data can be useful to detect even rumors of suspected outbreaks. As a result, more than 60% of the initial outbreak reports refer to such informal sources [WHO, 2015]. However, filtering this massive amount of data poses the difficulty to find the right criteria for which events to consider interesting and which to discard. The required high sensitivity to raise warnings about findings competes with the demand to filter the massive

amount of data from the web. At the RKI, two central units conduct EBS, namely EpiLag and INIG.

1.1.2 EpiLag

To have an adequate response mechanism that cannot be satisfied by existing surveillance infrastructure, in 2009 the **epidemiologische Länder-Bund-Konferenz** (*Epidemiological Federal State-State Conference*) (EpiLag) was established [Mohr et al., 2010]. In Germany, the health policy is handled by the federal states, and thus the infrastructure of the health authorities is different for each state. To still communicate efficiently, the EpiLag established a weekly conference call where the RKI and the federal states are exchanging on events of high importance for the public health which require immediate attention. The EpiLag makes recommendations on how to handle threats to public health and provides a platform to coordinate undertakings that affect several federal states.

1.1.3 INIG

The **Informationsstelle für Internationalen Gesundheitsschutz** (*The Information Centre for International Health Protection*) (INIG) is a young project that is staffed by epidemiologists with different backgrounds in medicine and public health to provide international epidemiological surveillance for the RKI. For this, they tasked epidemiologists with reading trusted sources (Tab. 4.1) for epidemiological articles to find international events that are particularly important for the public health of Germany. When INIG members are unsure whether an article describes a noteworthy outbreak, they have the opportunity to consult each other. This procedure often leads to a more educated decision about whether an outbreak article is interesting. INIG also supports the work of the EpiLag by providing intelligence about countries that exceed Germany's neighboring states, countries with which the RKI has no official information exchange agreement. According to former Minister of Health Hermann Gröhe, the importance of global health was most evident during the uprise of the Ebola disease and Zika epidemics starting in 2015 which showed that the protection of German citizens and the aid for affected people requires international cooperation.

1.2 Natural Language Processing and Epidemiology

Although the processing of informal sources as part of EBS promises faster and more effective surveillance, the large quantity of data and their inaccuracy poses the challenge to unlock the potential of those sources efficiently. Algorithms in the field of natural language processing (NLP), however, are well suited to tap these informal resources and

help to structure and filter this information. These algorithms are mostly data-driven, i.e., they independently select features from the data and are not necessarily dependent on formalized expert knowledge to return useful output. The general data-driven approach (bottom-up) in machine learning is perceived precarious in the health sector because it usually does not answer *how* a decision was made. Indeed, while a machine learning model could perform better than a designed formal model, the machine learning model's decisions used to be hard to disentangle. But, with the recent improvements in making those decisions accessible [Arras et al., 2017], and endeavors to tackle biases in machine learning models, those usual reservations can be better approached. Furthermore, with advancing digitalization of public health, more and better quality data become available [Benzler et al., 2014]. Due to that, a bottom-up approach is now more often a valid alternative.

This paradigm shift is particularly apparent in the field of NLP. Up to the end of the 20th century, mainly explicitly model translation algorithms were used, called **rule-based-machine-translation**. They demanded a behemoth of rules to account not only for two grammatical systems that needed to be aligned but also handle special cases, idioms, and dynamics of language [Bar-Hillel, 1953, 1960]. With the renaissance of deep learning, the bottom-up approach performed much better and required less modeling and at the same time accuracy in translations improved by a manifold [Bengio et al., 2003]. The ease with which already existing labeled data, like websites that provide their content in different languages, could be utilized was also exemplary for the advancing improvement of bottom-up methods [Macklovitch and Simard, 2000].

Signale is a group within the **Infectious Disease Data Science Unit** of the RKI that is mainly responsible for providing interfaces between application and machine learning, including NLP. Signale provides intuitive access to critical epidemiological analyses via dashboards for different use cases within the RKI and develops outbreak algorithms. Therefore, the Signale team is naturally interested in drawing from EBS and fueling the paradigm shift in epidemiological surveillance and utilizing the new capabilities of NLP. This thesis was written at the Signale team.

1.3 Motivation

The broad idea at the beginning of this project was to integrate NLP into a dashboard for epidemiologists who perform epidemiological surveillance to assist them in their intensive news survey. EBS was an exciting and well-suited topic for this project due to its dependency on large unstructured data. The RKI has two groups that perform EBS that would profit the most from NLP aided tools. The two groups, EpiLag and INIG, analyze numerous, also informal textual sources for their work and thus, I decided to look into

their work to pinpoint whether and how NLP might prove useful for them.

While the EpiLag focuses on disease outbreaks within Germany, INIG is mainly interested in international disease outbreaks. Since the EpiLag is a phone conference and the information provided to them is from the 16 state health departments of Germany, it is hard to retrace the origin of all their reported information. Due to the difficult access to their sources, they did not pose an ideal candidate for the development of an NLP-driven aid. INIG, however, is a project that depends much more on self-acquired intelligence. Unlike EpiLag, INIG collects information from well-defined, publicly accessible sources. Each week, one person of the INIG team reads articles from a fixed set of sources and filters out outbreak news that is considered important for the RKI, the ministry of health, or the German army. Then the INIG staff member fills an Excel sheet with the information from the found article. The overall process takes around 30 minutes every day which led to the idea to automate this process.

The first goal of the thesis was to automatically put key information like the disease or the number of confirmed cases from an outbreak article into a database and replace the previous Excel-workflow. Being able to describe the article based on its key information led to the second goal: The utilization of articles and their keywords to learn the relevance of an article and then use this knowledge to develop a recommendation system to decrease the burden of finding important articles. Third, when having a functional recommendation system, a further future goal was to unravel the epidemiologists' decision process and try to homogenize and standardize their workflow. With a working summarization and relevance scoring pipeline, a possible further goal was to expand the work of INIG to more sources and support the parsing of non-English text.

1.4 Related Work

The Global Rapid Identification Tool System (GRITS) by the EcoHealth Alliance is a website that allows evaluating epidemiological texts automatically. It extracts key information about the text and makes a classification about which disease is most likely thematized in the text. GRITS, however, is not automatable and customizable. In case INIG members would like to use GRITS, they would need to manually copy-paste URLs into GRITS and manually extract its output. Furthermore, GRITS does not filter news but only processes them. Hence, it does not reduce the burden of reading several outbreak articles.

MEDISYS is a webpage that filters and sorts outbreak news from a vast amount of sources of which INIG's used sources are mostly covered. However, the classification of disease outbreak news is untransparent and does not include key information extraction as in GRITS. It might filter for other criteria that are not of interest for INIG, and therefore

does not promise any time savings. Due to the high amount of sources listed at MEDISYS, the reading burden would be even increased.

CHAPTER 2

Background

2.1 Natural language Processing

Natural language processing (NLP) consists of a set of methods that operate on natural language (language that naturally evolved among humans) to solve tasks like translation, question answering, text summarization, and interaction with spoken language. While humans can intuitively solve these tasks, machines require a whole batch of preprocessing steps to utilize such raw data and cope with the redundancy in language and facilitate the otherwise not accessible underlying rules (including but not limited to grammar). Fig. 2.1 shows a pipeline that illustrates the language wrangling steps common in NLP for information extraction.

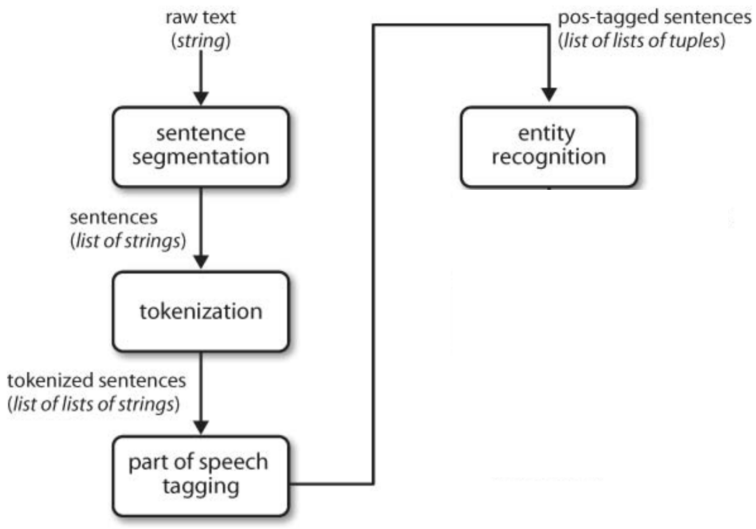

Figure 2.1: An illustration of a typical pipeline for information extraction in NLP. The pipeline starts with the raw text and undergoes several preprocessing steps (indicated by a rectangle) yielding intermediate output (placed next to the arrows). The final output can be a list of tuples of the entity and the corresponding word like `[(ORG, 'Bayer'), (LOC, 'Baveria')]` (adapted from Bird et al. [2009]).

2.1.1 Stop Words

Assuming we want to analyze texts on word level, we might be looking for words that appear more often than others or only appear in certain texts. This information can tell us a lot about the topic or sentiment of a text. Suppose we find that a text is frequently mentioning *Brexit* more than any other text. Then we can deduct that this text might be a (news) article and that it is about the exit of Great Britain from the European Union. However, we quickly realize that certain words also appear more frequently than others, without revealing much about the content of the text. Words like *the, of, than* appear numerously in every English text independently of the topic or the source. Such words first received particular attention when Luhn [1960] identified their property to obscure target words for further analyzes. These words are called **stop words**, and there are by now many curated lists of stop words for different languages and tasks [RANKS, 2019].

For any NLP algorithm that requires information about the grammatical structure of a text, we want to keep stop words since they convey a substantial part of grammatical information. However, should we be interested in the topic or source of the text, then it might be sufficient to search for this information within only a handful of words. Therefore, it is common practice to remove stop words to improve the performance of classification algorithms [McCallum and Nigam, 1998; Lodhi et al., 2002; Tong and Koller, 2001]. In state-of-the-art neural classifier like described by Howard and Ruder [2018], it is not always necessary.

2.1.2 Regular Expressions

Regular expressions (regex) is an expression using ASCII characters to define a set of strings that this expression matches. Regex consists of meta and literal characters. For a literal character holds that it matches this exact character in some target text. A metacharacter is interpreted and facilitates regexs [Kleene, 1951]. While these metacharacters vary between different regex libraries, most of them are identical. A literal character combined with a $*$ is a ubiquitous functionality (known as the Kleene star) and means that the literal character may appear 0 to n times in succession to allow a match. To still be able to use those metacharacters as literal characters, they can be escaped (generally with a backslash). Tokenization (2.1.3) or stemming (2.1.5) are preprocessing steps in NLP that sometimes use rules formulated as regexs.

2.1.3 Tokenization

A token is an abstraction of a piece of information. In NLP this can be a single character, word, punctuation, or sentence. The goal in **tokenization** is to split a text into meaningful

chunks that obey the rules of the natural language. Word tokenization is the cornerstone for the vast majority of NLP pipelines [Webster and Kit, 1992].

Tokens, however, do not always match how we think about words or sentences which can be shown by the following example. If we would formulate simple rules for tokenization, then a definition for a word would be a string enclosed by single white space characters. A simple regex for this rule would be `[a-zA-Z]+` that matches 1 to n (indicated by `+`) lower or uppercase letter (indicated by `[a-zA-Z]`) in succession. Words delimited by periods would then be a sentence. These regexs, however, do not always work as expected. The *United States of America* is such an example where our rules would fail. They lead to splitting this string into four tokens {'*United*', '*States*', '*of*', '*America*'} although these tokens are not independent of each other and thus should be one token. Longer names like this one often have acronyms such as the *U.S.A.* Following our simple rules, the word *U.S.A* would be split into six tokens {'*U*', '**.**', '*S*', '**.**', '*A*', '**.**'} since it is mistaken by a sentence.

On the other hand, using an established tokenizer is also wondering e.g, *don't* is split into {'*do*', '*n't*'} but we know it should be {'*do*', '*not*'} assuming tokens are a representation of words. However, since tokens are not meant to be a precise image of natural language words but rather a method to yield word level understanding then *not* represented as *n't* will not worsen any text analysis, if the usage of *n't* is consistent. So without a doubt, the *United States of America* or the acronym *U.S.A.* needs to be treated as a single unit for sufficient word-level understanding. It is common to use a curated list of expected abbreviations, names, and phrases to avoid bad tokenization. To improve sentence tokenization further, one could train an unsupervised sentence boundary detector as described in Kiss and Strunk [2006]. This method extracts the most occurring sentence breaks and learns to discard those that do not to seem to be a valid new sentence. The acronym *U.S.A.* will occur only as a fraction compared to more frequent sentence breaks. The most common one will be lower-case letters followed by a whitespace, a period, and then an upper-case letter in standard literature or new lines in online chats.

Sentence tokenization is based on word-tokenization and is also part of most text analyzes. It becomes essential, for example, if a text includes logical elements (like paragraphs or chapters) that need to be handled differently as in news summarization where the core content is more likely to be found in the first paragraph.

2.1.4 POS-Tagging

Although tasks like text classification perform already well with a text that only has been tokenized, mostly we require further processing to do more language-aware tasks. If the grammatical correction of texts is the goal of some NLP pipeline, then it goes without saying that we need more information about the grammatical function of the extracted

token.

We could, for example, want to identify all nouns from a text. This will be an easy task for many words like *car* or *mother*. They can be looked up in an English dictionary which will confirm that these words are nouns. However, words like *meeting* can take different parts of speech (verb or noun).

To find the noun, we apply **position-of-speech tagging** (POS-tagging). A POS-tagger is typically a machine learning model like a decision tree [Màrquez and Rodríguez, 1998] trained on an annotated corpus like Penn Treebank [Marcus et al., 1993], that already has the right grammatical entities assigned to each word in unstructured texts. When the correct grammatical function of all tokens is determined, it is simple to detect the grammatical number of the noun occurring before *that* and hence correct the sentence grammatically.

2.1.5 Lemmatization and Stemming

Having a well-tokenized text, we can do descriptive statistics on the text, e.g., by counting the occurrences of tokens or detect tokens that only occur once, so-called **hapax legomenon**. The simplest way to sensibly reason about a text would be to find the most frequent words, given that we removed stop words to infer the subject of a text. If the goal were to infer the type of sport based on the most occurring words in sports news we would hope to find a high term frequency for something like *throw* for baseball or *strike* for soccer. Merely counting tokens to display the most occurring ones in a text will not be sufficient. The words in the text will be used in many forms due to grammatical conjugation. For example, *thrown* and *throw* will be treated as unequal tokens by the computer although they mean the same. Therefore, we need to transform all words to their infinitive form.

There are two options: **stemming** and **lemmatization**. Stemming only prunes the end of words following rules. These rules are specified as regular expressions which try to exploit regularities in language to infer the infinitive form. However, they will not always reliably work due to special cases and ambiguities of natural language. One common rule among stemmers is the removal of *ing* at the end of a lower case word to transform a word into its infinitive form. This rule works correctly in most cases but fails for words like *lying* that the stemmer transforms to *ly* which is a not desired behavior.

Lemmatization is a more sophisticated method. First, it uses a grammatical database lookup instead of rules and therefore will be able to transform *lying* correctly to *lie*. Second, it incorporates POS-tagging to determine how words need to be transformed based on their grammatical function [Müller et al., 2015]. If the word *meeting* is used as a noun, we do not want to transform it to *meet* but keep it like it is. Lemmatization produces better results than stemming [Balakrishnan and Lloyd-Yemoh, 2014] for a higher cost of

computation.

2.1.6 (Disease) Named-Entity Recognition

Named-entity recognition (NER) is a step placed towards the end of an NLP pipeline (Fig. 2.1). After sentences and words have been tokenized, and POS-tagging was applied, it might be important for some learning algorithms to know the named-entity of words. Typical examples are the recognition of names of persons or companies, and numerical entities, like time, dates, and money [Nadeau and Sekine, 2009]. This recognition might be part of some NLP pipeline, but it also can be part of a pipeline for information retrieval (IR). The goal of IR is the extraction of specific information and their subsequent storage in a database to structure a large amount of data that is difficult to access [Manning et al., 2008; Wei et al., 2011].

The reason NER is not covered merely by a dictionary lookup but a separate module in a pipeline is the difficulty to deal with ambiguous words like *Apple*. When the word *Apple* is at the beginning of a text, it is unclear whether it stands for the fruit or a tech company. While in the beginning, NER consisted of handcrafted rules and heuristics [Jacobs et al., 1991], NER is now a classification problem that can be solved through supervised learning, capable of learning named entities with hidden Markov models, decision trees, or conditional random fields [Nadeau and Sekine, 2009]. Though, there is a shift towards un/semi-supervised methods that infer features (as neural networks) and outperform feature-engineered systems (as decision trees) [Yadav and Bethard, 2018].

In the medical field, such ambiguities rise not because the proper names are so indistinguishable from other common words, but because there are many forms how to write a disease name and equally many abbreviations (e.g., *cancer*, *carcinoma*, *malignant tumor*, *CA*). To reason from medical texts, it is necessary to identify those utterances that are most likely representing a disease. Thus, disease-NER is a crucial processing step in this domain. However, benchmarks also showed that in highly standardized, text dictionary lookups perform equally well [Jimeno et al., 2008]. Therefore, the exact procedure to perform disease NER depends on the source of interest.

2.1.7 Corpus

A **corpus** is an organized collection of (related) text documents that are simple to access. They can be annotated (as required for training POS-tagger) or preprocessed (e.g., tokenized) [Bengfort et al., 2018]. Preprocessing, however, can be time-intensive especially for large amounts of data and this process irreversibly changes the raw text. Thus, each intermediate step should be stored in the corpus as well. This way, quality control is assured, replacement of preprocessing steps is streamlined, and repeated computations or

even human labor is avoided. Optimally, a corpus is easily accessible and safely stored since it contains laboriously created data. NoSQL databases are advantageous for corpus saving due to their minimal overhead and allow access to data that would be too large for a normal machine. This way corpora can easily be shared by providing access to the database. Alternatively, they can be stored as a folder system containing `.txt`, `.XML`, or `.JSON` files.

2.2 Machine Learning

Machine learning (ML) is an attempt to make the computer learn. Mitchell described learning broadly as,

> "A computer program is said to **learn** from experience $\mathcal{E}$ with respect to some class of task $\mathcal{T}$ and performance measure $\mathcal{P}$, if its performance at tasks in $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$."

There are, however, ML algorithms that not necessarily define $\mathcal{E}$. Making experience from seeing labeled examples is called **supervised learning** and learning a model without labels is called **unsupervised learning**.

2.2.1 Bag-Of-Words

It is typical in a machine learning task that large amounts of data can make explicit feature engineering unnecessary. Also, instead of requiring too much domain knowledge, it is preferred to let the algorithm determine important features. One typical example of this in NLP is the **bag-of-words** approach.

It is particularly challenging to model language because of its entangled grammar, all its special cases, idioms, and ambiguity. Therefore, we can try to analyze a text based only on the occurrences of the words in the text without their grammatical context. We do so by tokenizing a text, lemmatize the tokens and operate on the set of words left after this preprocessing as our set of features.

A typical example of this simple approach is spam detection. Given a set of emails and labels *spam* and *not spam*, a machine learning model could learn to classify *spam*.

2.2.2 Imbalanced Class Problem

When it is required to detect seldom events, like cancer in patients through imaging, the performance of the classifier suffers from **imbalanced classes**. In the cancer example, this would mean that images showing a tumor are much rarer than those without cancer. A classifier then tends to classify most input as the majority class, i.e., the patient not

having cancer, which would yield a high accuracy but is fatal for the patient. The accuracy (the relative amount of correctly classified classes) in this case is not a useful measure, but there are many ways to tackle the class imbalance problem.

One way is cost-sensitive learning, which treats different misclassifications differently as opposed to cost-insensitive learning that only tries to maximize accuracy. Thus, in cost-sensitive learning, it is possible to explicitly model the, e.g., minimization of false negatives such as in the cancer detection example by introducing a cost function for misclassifications or shift the decision threshold of the classifier towards the imbalanced classes [Ling and Sheng, 2008].

Another way is using up- and downsampling methods. A naive downsampling approach would be to discard data of the majority class until the classes are balanced. The naive counterpart to downsampling is the duplication of minority class instances and is called upsampling. However, there are more sophisticated methods, that will downsample redundant majority class instances or synthesize data, in order to create novel examples of the minority class. One well performing upsampling method is called **Adaptive Synthetic Sampling Approach for Imbalanced Learning** (ADASYN) [He et al., 2008]. The general procedure of ADASYN is described as,

$$d = \frac{m_s}{m_l} \tag{2.1}$$

$$G = (m_l - m_s) \cdot \beta \tag{2.2}$$

$$r_i = \frac{\Delta_i}{K}, i = 1, \ldots, m_s \tag{2.3}$$

$$\hat{r} = \frac{r_i}{\sum_{i=1}^{m_s} r_i} \tag{2.4}$$

$$g_i = \hat{r} \cdot G \tag{2.5}$$

$$\forall \boldsymbol{x_i} \text{ repeat } g_i \text{ times: } \boldsymbol{s_i} = \boldsymbol{x_i} + (\boldsymbol{x_{zi}} - \boldsymbol{x_i}) \cdot \lambda \tag{2.6}$$

where d is the degree of imbalance of the small and large class (m_s and m_l respectively), G is the number of synthetic data examples required to achieve a class balance of $\beta \in [0, 1]$. Δ_i is the number of examples having K nearest neighbors of $\boldsymbol{x_i}$ from the majority class, (2.4) is a normalization step, and g_i is the number of synthetic data instances that need to be generated for each minority example $\boldsymbol{x_i}$. (2.6) is the main algorithm where for each data point $\boldsymbol{x_i}$, g_i synthesizations are conducted. Thereby, $\boldsymbol{x_{zi}}$ is a random data example from the K nearest neighbors, and $\lambda \in [0, 1]$ is a random number.

Finally, a good performance measure is required to evaluate the classifier's learning progress. Accuracy, as illustrated before, is not a reliable indicator in an imbalanced two class dataset. The **Index of Balanced Accuracy** (IBA), however, tries to take the

imbalance into account and is defined as,

$$IBA_\alpha = (1 + \alpha \cdot (TPrate - TNrate)) \cdot TPrate \cdot TNrate$$

where $0 \leq \alpha \leq 1$ and α being a dominance index, that needs to be fine-tuned based on how significant the dominating class is supposed to be.

2.2.3 Naive Bayes Classifier

The **naive Bayes classifier** (NBC) is a probabilistic classifier. It describes a set of algorithms capable of learning to infer a class given a set of features and labels. The NBC is defined as,

$$\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\} \tag{2.7}$$

$$\mathcal{C} = \{\mathcal{C}_k \mid k \in K\} \tag{2.8}$$

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\boldsymbol{x})} \tag{2.9}$$

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(x_1|\mathcal{C}_k)p(x_2|\mathcal{C}_k)\ldots p(x_n|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\boldsymbol{x})} \tag{2.10}$$

where $\boldsymbol{x}$ is a data point which consists of n features. $\mathcal{C}$ is the set of all classes, and $p(\mathcal{C}_k|\boldsymbol{x})$ is the probability $\boldsymbol{x}$ belonging to class $\mathcal{C}_k$. We are allowed to write (2.9) as (2.10) due the independene assumption of the features of $\boldsymbol{x}$,

$$p(x_i|\{\forall x_j \in \boldsymbol{x} : j \neq i\}, \mathcal{C}_k) \overset{x_i \perp\!\!\!\perp \forall x_j}{=} p(x_i|\mathcal{C}_k)$$

A standard procedure to predict the most likely class that $\boldsymbol{x}$ belongs to, is to find the maximum a posteriori probability

$$\arg\max_{k \in \mathcal{C}} p(\mathcal{C}_k|\boldsymbol{x}) \tag{2.11}$$

To evaluate (2.10), we need to take the product of the probabilities of the features given the class and then consider the class with the highest probability for the classification as in (2.11).

Note, the classifier is named *naive* because we assume that every feature in the vector $\boldsymbol{x}$ is independent, i.e., there is no correlation between them. This assumption is most of the time wrong, but in practice, NBC still performs very well [Rish, 2001].

2.2.3.1 Multinomial Naive Bayes for Text Classification

For classifying texts based on the words they contain as in the bag-of-words approach, we can apply multinomial naive Bayes as follows,

$$\mathcal{C} = \{\mathcal{C}_k \mid k \in K\} \tag{2.12}$$

$$\boldsymbol{t} = \{c_1, c_2, \ldots, c_n\} \tag{2.13}$$

$$\boldsymbol{T}_k = \{\boldsymbol{t}_d | \forall d \in \mathcal{D}_{\mathcal{C}_k}\} \tag{2.14}$$

$$p(t_{k,i}|\mathcal{C}_k) = \frac{t_{k,i}}{\sum_{\forall t_j \in \boldsymbol{T}_k} t_{k,j}} \tag{2.15}$$

$$p(\mathcal{C}_k|\boldsymbol{t}_d) \propto p(\mathcal{C}_k) \prod_{i=1}^{|\boldsymbol{t}_d|} p(t_{d,i}|\mathcal{C}_k) \tag{2.16}$$

Where $\mathcal{C}$ is the set of text classes, and $\boldsymbol{t}$ contains the token counts of the whole vocabulary, c_i being the occurrence count of term i. $\boldsymbol{T}_k$ is a matrix where the token counts c are the columns, and the documents d the rows consisting of all documents $\mathcal{D}$ from class $\mathcal{C}_k$. To calculate the likelihood of $p(t_{k,i}|\mathcal{C}_k)$ we divide the occurrence of token $t_{k,i}$ by the sum of all token occurrences in the same class $\mathcal{C}_k$. Following the independency assumption as in (2.10) we now can calculate $P(\mathcal{C}_k|\boldsymbol{t}_d)$ as the product of the prior $P(\mathcal{C}_k)$ – the probability of class $\mathcal{C}_k$ as learned from the training set – and the probabilities of all the terms $t_{1\ldots n}$ given the prior class. For classification, the arg max as in (2.11) is taken.

Three problems can occur in text classification that need to be handled. First, the high probability of numerical underflow due to the large product of several values < 1, second, some token $t_i \in \boldsymbol{t_d}$ having a count of 0 that would nullify the whole product, and third, large disparities of token counts that bias the classification. The solution to the underflow problem is to transform (2.16) into log-space which equals to:

$$log(P(\mathcal{C}_k|\boldsymbol{t}_d)) \propto \log(P(\mathcal{C}_k)) + \sum_{i=1}^{|t_d|} \log(p(t_{d,i}|\mathcal{C}_k))$$

To avoid $p(t_i|\mathcal{C}_k)$ becoming 0 we apply Laplace smoothing to (2.15) which yields

$$p(t_{k,i}|\mathcal{C}_k) = \frac{t_{k,i} + 1}{\sum_{\forall t_j \in \boldsymbol{T}_k} t_{k,j} + 1}$$

and to avoid a bias towards some exceptionally frequent words, **term frequency-inverse document frequency**(tf-idf) corrects this discrepancy by weighting the term frequency

as follows,

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.17}$$

$$\mathrm{idf}(t,d,\mathcal{D}) = log\left(\frac{|\mathcal{D}|}{|d \in \mathcal{D} : t \in d|}\right) \tag{2.18}$$

$$\text{tf-idf}(t,d,\mathcal{D}) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,d,\mathcal{D}) \tag{2.19}$$

where $f_{t,d}$ is the term frequency of term t in document d and t' is any term in d. $|\mathcal{D}|$ is the number of documents in the corpus and $|d \in \mathcal{D} : t \in d|$ is the number of documents containing term t. (2.19) is the final formula that balances discrepancies between term frequencies over several documents.

2.2.3.2 Bernoulli Naive Bayes

Especially for shorter texts and binary decision (*spam*, *not spam*), there is only a small gain to use token counts. The Bernoulli variant of the NBC explicitly models the absence and presence of words which is done by a different approach to estimating $p(t_i|\mathcal{C}_b)$ (comparison to 2.15). The probability of token t_i given some binary class is calculated like,

$$p(t_i|\mathcal{C}_b) = p(t_i)(1 - p(t_i))^{(1-b)}$$

where $b \in \{0,1\}$ and $\mathcal{C}_b$ is the corresponding class.

2.2.3.3 Complement Naive Bayes

There are cases where the *naive* assumption is particularly ill made. In an unbalanced class problem, the data is skewed towards the larger class which the following table illustrates,

Table 2.1: A statistical evaluation of a coin flip experiment with an imbalanced class. θ is the probability of a class to yield head (H). We flip one coin in Class 1 and two coins in Class 2 per row. p(*data*) is the probability for the seen coin flip. The column Label for H contains the class assignment after the coin flip (adapted from Rennie et al. [2003]).

Class 1 $\theta = 0.25$	**Class 2** $\theta = 0.2$	**p(data)**	**Label for H**
T	TT	0.48	*none*
T	{HT, TH}	0.24	$Class2$
T	HH	0.03	$Class2$
H	TT	0.16	$Class1$
H	{HT, TH}	0.08	$Class1$
H	HH	0.01	*none*

The experiment shown in Tab. 2.1 , that considers each possible outcome of coin

flips, suggests a 24% probability for Class 1 to yield H and 27% for Class 2, although the probability for Class 1 to yield H is actually higher. The proposed solution by Rennie et al. [2003] is to minimize the probability of a vector of word counts **not** belonging to a class to tackle class imbalance,

$$\arg\min p(\neg \mathcal{C}_k) \prod_{i=1}^{|t_d|} \frac{1}{p(t_{d,i}|\neg \mathcal{C}_k)}$$

2.2.4 Support Vector Machine

A prominent application of vector space based machine learning is the **support vector machine** (SVM). It is a classification algorithm that is trained in a supervised fashion. It is often used for document classification and performs well even on small data [Manning et al., 2008]. During training time, the SVM tries to find a decision boundary in the vector space of the training data to separate two classes with a maximal margin. This is accomplished by choosing a hyperplane that has a maximum distance to the closest data points of both classes respectively. The margin is defined by those points which are called support vectors (Fig. 2.2). For classification, we calculate $\mathrm{f}(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + b)$ where $\mathbf{w}$

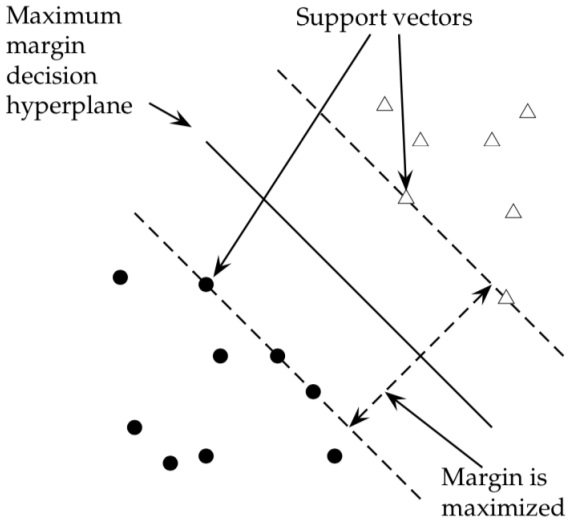

Figure 2.2: An illustration of how support vector machines realize classifications. Black dots and white triangles represent two classes of data points (adapted from Manning et al. [2008]).

is a weight vector orthogonal to the decision hyperplane, $\mathbf{x}$ is the input data point, and b an intercept. Training an SVM is an optimization problem which can be solved with, e.g., the Lagrange multiplier, denoted as α_i. Then,

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

$$b = y_k(1 - \zeta_k) - \mathbf{w}^T\mathbf{x}_k \text{ where } k = \arg\max_k \alpha_k$$

where y_k is the label of data point $\mathbf{x}_k$ and all $\alpha_i \neq 0$ are the support vectors. For all α it hols that $0 \leq \alpha \leq C$, where C is a regularization term to control overfit. ζ is a slack variable that allows the classification with a soft margin, i.e., drawing a margin that classifies some training data points incorrectly for a penalty equals to ζ. The classification function then is $\mathrm{f}(\mathbf{x}) = \mathrm{sign}(\sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b)$. There, however, exists much faster, more scalable solutions for training SVMs that I will not thematize.

Until now we were only able classifying linearly separable classes. Unfortunately, classes are mostly not linearly separable in document classification. A solution to this is to apply the *kernel trick* to find a higher dimensional space in which the data points are linearly separable. To do so, we use a function $\Phi : \mathbf{x} \mapsto \phi(\mathbf{x})$. It suffices to only calculate the result of the dotproduct $\phi(\mathbf{x}_i^T)\,\phi(\mathbf{x})$ in this higher dimension space. Φ is called the kernel function of which several exist. One popular kernel function is the **radial basis function** which is defined as

$$\Phi_{\mathrm{RFB}}(\mathbf{x_i}, \mathbf{x}) = \exp(-\gamma(\mathbf{x_i} - \mathbf{x})^2), \gamma > 0$$

with γ being the kernel function coefficient.

2.2.5 Logistic Regression

The logistic regression, similar to linear regression, is commonly used for classification in ML. It uses the logistic function, also called sigmoid function and is defined as,

$$\sigma(\mathrm{x}) = \frac{1}{1 + e^x}$$

that is used to map the output of a linear model to $\hat{y} \in [0, 1]$,

$$\hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

where the weights $\mathbf{w}$ are adjusted during training to minimize

$$\ell(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

with y being the true label, $\mathbf{x}$ the n-dimensional input, and b a bias value.

The loss can be minimized using gradient descent (as used in backpropagation in 2.2.7.2). Usually, more optimized methods are used.

To avoid overfitting, we can apply a penalty, for example, L2 which adds a term dependent on $\mathbf{w}$ to the loss ℓ to force the model to choose smaller weights. L2 is defined as $C \sum_{i=1}^{n} w_i^2$ with C being the regularization strength where larger values of C increase

the penalty.

The logistic regression classifier in NLP is used to classify based on embeddings, e.g., document classification where the document embedding is used as the input to train the logistic regression. Note, the output $\hat{y}$ is not strictly binary; thus, we need to apply a threshold where typically values above 0.5 are classified as 1 and 0 otherwise.

2.2.6 k-Nearest Neighbor Classifier

The k-nearest neighbor classifier performs instance-based learning which means it has no training time. It remembers all training data points and then classifies a new input based on its k-nearest neighbors, where each data point $\mathbf{x}_i$ is a vector with a label y_i and $k \in \mathbb{N}$. The closest points can be calculated using several metrics. Most often, Euclidian distance ($\|\mathbf{x}_i - \mathbf{x}_j\|_2$) is taken to find the k nearest neighbors for classification. Closer points can be weighted stronger for the classification. The predominant label of the k-nearest neighbors (with or without weighting) is taken to be the classification of the new input. Typically, a small k is less robust against noise while a large k might not be specific enough.

2.2.7 Deep Learning

The idea to model the brain led to the development of the perceptron, which is a formal and simplified replica of a biological neuron. With the development of the backpropagation algorithm, it was then possible to stack several perceptrons and develop the multilayer perceptron which was capable of image and vowel classification [Russell and Norvig, 2009], also referred to as neural networks. For a long time, training these models was computationally expensive and thus limited to small-sized neural nets. This long believed limit changed only recently with AlexNet [Krizhevsky et al., 2012] which used graphics cards to stem the massive computational demand to train deep neural networks.

2.2.7.1 Perceptron

A **perceptron** is an algorithm inspired by biological neurons. It is a binary classifier that learns by adjusting its threshold to *fire* an *action potential* when the correct class is detected. Formally it is

$$\text{Perceptron}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

where $\mathbf{x}$ is a data vector, $\mathbf{w}$ the weight vector and b the bias. It is trained in a supervised fashion. The learning step with which w is adjusted is defined as,

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta(d(t) - y(t))\mathbf{x}(t)$$

with d being the desired output and y the output of the perceptron at training step t with a learning rate $\eta \in (0, 1]$.

2.2.7.2 Multilayer Perceptron

A **multilayer perceptron** (MLP) (Fig. 2.3) consists of three types of layers, an input and output layer, and $1 \dots n$ hidden layers. Each layer consists of an arbitrary number of perceptrons. The activation function is not a Boolean function but $\sigma(y) = \frac{1}{1+e^{-y}}$ called the sigmoid function. Its output also ranges from 0 to 1, but it is derivable. Derivability is crucial since the MLP uses backpropagation to be trained which is defined as,

$$E(\mathbf{w})_t \equiv \frac{1}{2}\sum(\mathbf{d}_t - \mathbf{y}_t)^2 \tag{2.21}$$

$$\Delta\mathbf{w} = -\eta\frac{\delta E_t}{\delta \mathbf{w}} \tag{2.22}$$

where E is the sum over the errors over all output neurons at step t and $\Delta\mathbf{w}$ is the weight adjustment according to the backpropagated error $\frac{\delta E_t}{\delta \mathbf{w}}$ that is the partial derivative of the error given all the weights with a learning rate η. The minus sign indicates that the weight is adjusted downwards the estimated gradient (minimization).

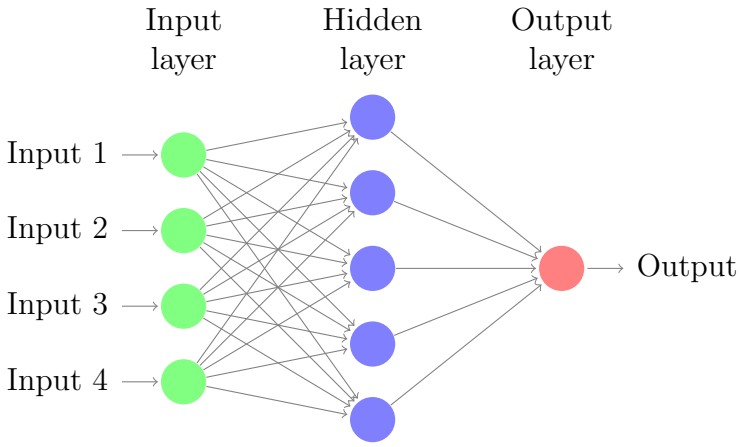

Figure 2.3: A illustration of a multilayer perceptron. The input (green), hidden (purple), and output (red) layer consists of four, five, and one perceptron respectively. The arrows indicate the direction of the computation of the input which is a scalar.

2.2.7.3 Convolutional Neural Network

While the MLP is only capable of receiving a single input vector at a time, the **convolutional neural network** (CNN) can process an input matrix. The additional dimension can then represent time or spatial dependencies. So-called feature maps extract these two-dimensional features in the hidden layers. Feature maps are usually smaller than the input matrices, and they consist of real-valued weights that are adjusted during the training process. These feature maps are striding over the input and apply a convolution,

$$C(i,j) = (I * F)(i,j) = \sum_m \sum_n I(m,n)F(i-m, j-n)$$

where $*$ is the convolutional operator, I is the input matrix, F the feature map, and $C(i,j)$ the output for the convolution at position (i,j) of the feature map. The output matrix C of the convolution step is then pooled which is, i.e., the maximum or average value of C. In the end, a fully connected layer (an MLP where every neuron is connected with each neuron) incorporates all information and yields a classification (Fig. 2.4).

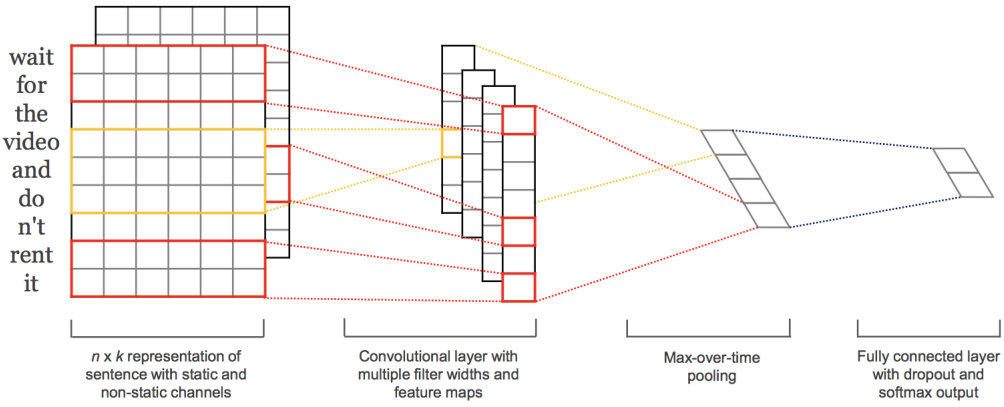

Figure 2.4: An illustration of an n-gram convolutional neural network. The input is a numerical representation (2.2.8) of a tokenized text. The convolution over the input matrix is represented as a red and yellow box (called feature map) that is then pooled (maximum of the kernel). The classification is then enforced by a fully connected layer that incorporates the input of all feature maps.

2.2.8 Embeddings

In 2.2.1, I introduced the bag-of-words approach to cope with the difficulty to explicitly model language, and in 2.2.3 I showed that this approach even facilitates a good performing classifier. However, the bag-of-words approach models language incorrectly and having a better representation of language is more desirable for the improvement of ML algorithms. **Word embeddings** are such methods that model language more accurately

by representing words as n-dimensional vectors that much better capture syntactic and semantic characteristics of language.

2.2.8.1 Word2Vec

In 2013, Mikolov et al. published two algorithms to compute high quality distributed representations of words and phrases efficiently. Both these algorithms are also referred to as **word2vec**. The general approach in both algorithms is to maximize the similarity measures of vectorized words that appear in a similar context. The **continuous bag-of-words model** is trained to predict the vector representation of a word or phrase given n words before and after the target word in its sentence. The **continuous skip-gram-model** is a slower implementation with the benefit to model uncommon words better [Code Google, 2013]. The skip-gram-model tries to predict the context given a target word and is thus the opposite approach of the continuous bag-of-words model.

In practice, we generate labeled data in the form of tuples $(\mathbf{w}_t, \mathbf{w}_{c_1}, \ldots, \mathbf{w}_{c_n})$, where one entry is the target word $\mathbf{w}_t$, and the other entries are context words $\mathbf{w}_c$. The tuple can either contain actual context words found in the text, or random words from the vocabulary (**negative sampling**). The embedding layer learns the weights of the input and context words, by yielding an n-dimensional vector for all these words, calculates the dot-product of the target and context vectors in the merge-layer, and then passes it to a sigmoid layer that predicts whether the words are actual context words or were negatively sampled. The objective of the algorithm is to maximize

$$\sum_{t \in T} \sum_{c \in \mathcal{C}_t} log(p(\mathbf{W}_c | \mathbf{w}_t))$$

where $\mathbf{w}$ is the vector representation of a word, t is the target word and $\mathbf{W}_c$ the context matrix, where each row is a vector representation of a context word.

2.2.8.2 GloVe

Word2vec embeddings are local since only n surrounding words are considered to calculate the embedding. **GloVe**, on the other hand, also tries to incorporate global information about word occurrences.

As the first step, a word co-occurrence matrix over all documents is calculated. The underlying assumption of GloVe is that the co-occurrence ratio is connected to meaning. Let X_{ij} be the co-occurrence count of token j in the context of i. Let $X_i = \sum_{t \in T} X_{it}$ be the occurrence of token i given any other token then $p(j|i) = \frac{X_{ij}}{X_i}$ is the probability of token j appearing in the context of token i. To be able to model the semantic information of words, their relationship needs to be modeled. GloVe defines a function $F((\mathbf{w_i} - \mathbf{w_j})^T \widetilde{w}_t) = \frac{p(i|k)}{p(j|k)}$ where $\mathbf{w}$ is a real-valued word vector. Hereby, the context words are subtracted from

the input words, and the dot product with the weights of the output vocabulary is taken (see Pennington et al. [2014] for details). This step facilitates embedding arithmetic, i.e., the meaning of word embeddings can be deducted from arithmetic calculations such as $w_{man} + w_{royalty} = w_{king}$. Further steps are required for computability, such as weighting of the word vectors or handling zero entries which are described by Pennington et al. [2014].

2.2.8.3 Document Embedding

If an algorithm only needs to operate on the document level, then the whole document can be embedded. A simple approach is to take the mean or the maximum of all word embeddings of a document. There are also more elaborated methods [Wu et al., 2018; Liu et al., 2018; Dai et al., 2015] that have a learning objective, similar to word embeddings. However, their downside is the demand for a large corpus to learn a meaningful **document embedding** which is not available in the scope of this thesis.

2.2.9 Web Scraping

The process of automatically extracting content available on the world wide web is called **web scraping**. It is different from web crawling, where the primary purpose is the following of hyperlinks on websites to index the linkage between pages. However, both techniques can be combined to systematically search for some specific content within a network of websites, e.g., crawling flight provider sites and monitor their prices.

2.2.9.1 HTML and CSS

Hypertext Markup Language (HTML) is a markup language which means that it consists of a set of *tags* that define how the content of a document needs to be interpreted. In HTML, for example, `<h1> I am a Header </h1>` defines the opening of a header tag followed by some content and the closing of this tag. When this HTML code is interpreted, the text is then displayed bold and larger than non-header text if not specified otherwise.

Cascading Style Sheets (CSS) is a style sheet language that is used for styling HTML code. Given the example above, we could modify the header to have another color, `<h1 style="color:red"> I am a Header </h1>`. It is also possible to define a **CSS class** and use this class to automatically apply several styles such as in Lis. 1, This way,

```
.header {
  color: red;
  font-family: verdana;
}
```

Listing 1: CSS class named *header* that sets the font to verdana and color to red when used.

we can apply the style like so `<h1 class="header"> I am a Header </h1>`. While a

class can be accessed from several HTML tags, there is also a **CSS id** that can only be accessed once in one HTML file but otherwise has the same functionality as a CSS class. Both languages thereby reveal a way of filtering the content of a website. Web scraping is exploiting HTML tags and CSS selectors like classes or ids to filter the content of a site.

Important tags for scraping are the paragraph tag `<p>` that normally contains text intended for the reader of a website and the `<a>` tag that contains hyperlinks. By investigating websites thoroughly, one might also find a schema in the assignment of CSS selectors that can be exploited during web scraping.

2.2.9.2 REST

Most websites offer a web service which is a broad term for some user interface that allows the communication to a database and optionally performs some operation on this data. The usual implementation of an application program interface is the **Representational State Transfer** (REST), and its methods are GET, POST, PUT, PATCH, and DELETE which are often executed via HTTP. The most important method in web scraping is the GET method since it retrieves data. It is sometimes necessary to understand how GET requests are made on a website to use them for automated content extraction.

2.2.9.3 Ajax

Should the client (user) request data from a database via a website and a reload of the site is undesired, **Ajax** allows an asynchronous data retrieval. This asynchronous GET request starts on an event (e.g., clicking on a button on the website). Ajax is frequently used to reduce loading time and traffic to only refresh the necessary part of a website. One framework to use Ajax is **jQuery** which is mostly used in combination with JavaScript that is required to integrate the retrieved data into the website.

A problem of scraping and crawling programs is that content, which is only visible after Ajax has requested it, is not visible to the program. Therefore, it might be necessary to monitor the website's behavior via the developer mode of a web browser and see which GET requests would retrieve the data of interest or trigger the Ajax calls programmatically.

CHAPTER 3

Methods

3.1 Requirement Analysis

At the beginning of my work, I needed to assess the suitability of several possible targets at the RKI for NLP. Therefore, if applicable, I monitored the functioning of the target groups and evaluated the text sources they used. The direction of the thesis was determined by the possible gain for their work practices through NLP and the exploitability of their used sources.

Below, I introduce available targets, and why they were or were not suitable for the proceedings of this thesis.

3.1.1 RASFF

The Rapid Alert System for Food and Feed (RASFF) has been established by the European Union (EU) to share information for effective food safety. As a member of the RASFF, the RKI receives PDFs about recent incidences of contaminated food. Such a PDF contains several prototypical information about the contamination.

However, the documents are strongly formatted, and my attempts to extract the text from the PDFs without altering the structure of the report failed. I used Tika (3.2.7) for the text extraction, but the return contained misplaced line breaks and other formatting errors. Due to these complications and difficulties to extract text from these PDFs without optical character recognition, I decided not to work with RASFF reports for my thesis.

3.1.2 EpiLag

Due to the majority of communication in the EpiLag happening orally and the sources mainly being reports from medical practices, hospitals, and emails, EpiLag poses an

ambitious target for NLP methodology. Without a clear in- and output, there is no clear approach to process the data of EpiLag.

3.1.3 EPIS

As the RASFF, the Epidemic Intelligence Information System (EPIS) is also an EU project. It is a web tool that allows several EU members to report possible or confirmed disease outbreaks. To stay up-to-date, EPIS has an email notification system, and every outbreak report is downloadable as a formatted Excel sheet.

The downside of EPIS is a missing source for reported outbreaks. Mostly, local health departments detect these events which are then shared through EPIS leaving out how they were discovered. This procedure makes the comparison of events difficult especially when different countries do not share the same definition for a disease outbreak. Though the formatted data output of EPIS would have been ideal for my further work, the regulations on how to process EPIS data were unclear and partially restrictive which excluded to further work with EPIS data.

3.1.4 INIG

INIG has a curated list of sources that they visit every day and check for new alerting events (Tab. 4.1). The list consists of a various set of sources where some are frequently reporting and some less but in more extent. Also, the data format is different and can be a simple HTML website, PDF, or email. The rather easy and public access to the used sources, the clearly defined in- and output, and necessary active acquisition of information as described in EBS makes INIG a good fit for my topic.

3.2 Libraries

In the following sections, I am introducing the most important libraries I used for my work. The list lacks libraries that are common for data-driven work with Python or are part of the standard library.

3.2.1 NLTK

The Natural Language Tool Kit (NLTK) is an established and large package for NLP. It has 50 corpora and lexical resources as well as all important algorithms to build an NLP-pipeline as in Fig. 2.1. NLTK was built initially to teach NLP and thus, contains also outdated algorithms in the field of NLP.

I used the stop word list, and the word and sentence tokenizer provided by NLTK. Furthermore, I used the multinomial NBC in NLTK to investigate the features with the highest explanatory power for classification, i.e., words that influence the classification decision by the NBC the most.

3.2.2 EpiTator

EpiTator is an epidemiological annotation library that is mainly used by GRITS. I used the following features of EpiTator in my thesis:

- EpiTator's count annotator that extracts the glyph and word representation for numbers and associated attributes, e.g., *"5 cases of smallpox"* will detect *5* as a count and *cases* as its attribute
- The date annotator that extracts dates and date ranges from a text
- EpiTator's geoname annotator that extracts all geographical entities from a text
- Finally, I also used Epitator's resolved keyword annotator, that uses an SQLite database of entities to detect disease entities from multiple synonyms of infectious

3.2.3 SpaCy

SpaCy is an industrial NLP library that contains every basic algorithm as illustrated in 2.1, but also newer methods from the field of deep learning such as CNNs and word embeddings. Additionally, SpaCy is written in the much faster CPython language. Altogether, speed and a selection of only the best-performing algorithms for NLP tasks give SpaCy the industrial strength.

I used SpaCy's text classification module which uses a simple CNN. SpaCy is also mainly used by EpiTator for preprocessing. Saving the output of EpiTator was difficult because SpaCy is not trivial to serialize. Therefore, I needed to transform the output of EpiTator such that it did not contain SpaCy dependencies anymore.

3.2.4 Flair

Flair is a small library that only contains state-of-the-art algorithms in NLP. It does not directly provide preprocessing such as stop word removal, tokenization, or stemming but does the preprocessing automatically depending on the final task.

Flair has a powerful word embedding module with a larger array of customizations which I used for the text classification pipeline.

3.2.5 Beautiful Soup

Beautiful Soup is a package to parse HTML and XML as a tree structure that can be traversed. Given the structured access to HTML content via tags, classes or ids, web scraping is much simpler with Beautiful Soup. I performed web scraping solely with Beautiful Soup.

3.2.6 Boilerpipe

Boilerpipe is a Java package that uses shallow text features (word count, hyperlink density, and position of text block) to remove *boilerplate* (advertisement, navigation bars,...) from websites [Kohlschütter et al., 2010]. There exists a Python package that calls the Boilerpipe Java code from a Python environment.

Although it would have been possible to write the web scrapers such that they would only extract the main content, I wanted to have a solution that allows expanding the work of my thesis to more websites without the necessity to write a special scraping program for each site. Thus, I wrote scrapers that extract the whole HTML content of a site containing some outbreak article and then used Boilerpipe to obtain the main purport.

3.2.7 Apache Tika

Apache Tika is a broad content analysis framework for text and metadata extraction from over a thousand data types that I used for text extraction from PDFs.

3.2.8 Luigi

Luigi is a pipeline builder made by Spotify. It manages different jobs, coordinates dependencies, and visualizes them. With Luigi, I modularized my work in case there is a necessity to modify or replace certain parts of my pipeline. Furthermore, Luigi automatically serializes the outcome of long and tedious computations such as scraping.

3.2.9 Flask

Flask is a web development mini-framework for Python. To demonstrate the final product which is capable of extracting the main content from some URL, annotating and evaluating this content, and then putting this information into a database, I built a web app using Flask.

3.3 Data Acquisition

After the decision was made to focus on data associated with INIG, I pinpointed the most important sources used by INIG, namely WHO DONs and ProMED Mail (4.1.1), to build a labeled dataset. In the following, I show the necessary data required for the labeling, building, and preprocessing of the dataset.

3.3.1 Incident Database - *Ereignisdatenbank*

The **Ereignisdatenbank** (EDB) is an Excel sheet (Fig. 3.1) and the primary recording method of the work of INIG. They track various parts of their work in this sheet but most

Ausgangs- bzw. Ausbruchsland	Sekundär betroffene Länder*	Erreger	Krankheitsbild(er)	Frühestbekannter Ereignisbeginn	Erstveröffentlichung	Fälle gesamt*	Datenstand für Fallzahlen gesamt*	Fälle bestätigt	Warchei nlische Fälle	Verdachtsfälle	Todesfälle	Quelle 1*
Vereinigtes Königreich		MERS-CoV	MERS	16.08.2018	23.08.2018	1	43335	1				ProMED-mail
Italien, Griechenland, Rumanien, Ungarn, Frankreich	Israel, Serbien	West-Nil-Virus	West-Nil-Fieber	Trend	Woche vom 17. bis 23. August	136	43336				19	ECDC RT Report
Afghanistan, DR Congo Nigeria Somalia		cVDPV2			Woche vom 17. bis 23. August	4 cVDPV2 2 WPV1 in Afghanistan	43335					GPEI
Kanada		Listeria monocytogenes				0						FoodSafetyNews
USA, Delaware		Lyssavirus	Tollwut			1					1	OutbreakNewsToday
Vereinigtes Königreich		MERS-CoV	MERS	13.08.2018	23.08.2018	1	43335	1				ECDC RRA
Kongo		Gelbfiebervirus	Gelbfieber		28.08.2018	1	43340	1		186		ProMED-mail

Figure 3.1: A screenshot of a part of the Ereignisdatenbank (*Incident Database*) Excel spreadsheet.

importantly they enter every critical outbreak article into the EDB. Every entry consists of several columns, of which only some are mandatory such as the reported **disease**, the **country** of origin, the number of **confirmed cases**, and the case number's reporting **date**. All WHO DON and ProMED Mail articles and the corresponding key information placed in the mandatory columns of these articles serve as training examples for the later mentioned classification algorithms.

3.3.2 WHO DONs

The WHO regularly publishes the latest disease outbreak news (DONs) as a publicly available web resource. It is a low output source with only a handful of reports every week. All WHO DONs are archived, sorted by year and month, and therefore can be systematically accessed.

I wrote a scraper that accesses the archive URL and then, based on the function parameter, visits the URL with the requested time range and scrapes the content (Lis. 2).

```
# Obtain annual report archive links
page = requests.get('http://www.who.int/csr/don/archive/year/en/')
soup = BeautifulSoup(page.content, 'html.parser')
archive_years = soup.find('ul', attrs={'class': 'list'})
all_years_links = archive_years.find_all('a')
years_as_links = ['http://www.who.int' + link.get('href')
                  for link in all_years_links]

# Obtain all report URLs per year
for year_link in years_as_links:
  page_year = requests.get(year_link)
  soup_year = BeautifulSoup(page_year.content, 'html.parser')
  archive_year = soup_year.find('ul', attrs={'class': 'auto_archive'})
  daily_links = ['http://www.who.int' + link.get('href')
                 for link in  archive_year.find_all('a')]
```

Listing 2: An extract from the WHO DONs scraping script usgin BeautifulSoup. The algorithm starts with extracting the content of '`http://www.who.int/csr/don/archive/year/en`', then filters the URLs for those referencing archived reports sorted by years with the help of the `ul` tag and `list` class. To extract all DONs per year, the `auto_archive` class is used. All links are found in the `a` tag and `href` selector.

3.3.3 ProMED Mail

In contrast to WHO DONs, much more authors are contributing to ProMED Mail and generate higher output. Usually, ProMED publishes around a handful of disease outbreak articles a day. ProMED mail content is dynamically loaded via Ajax and therefore not directly accessible. Through the analysis of the website, I reverse engineered the article search of the website to write a function with which I can scrape ProMED article given a time range shown in Lis. 3. Access to the page number (shown in Lis. 3) is necessary since after 200 pages the GET request returns an error although there would be more pages. The full algorithm iterates over 200 pages given a time range, remembers the date of the oldest ProMED article from this search and reruns the search with the initial temporal lower bound and the date of the last retrieved article as the newer upper bound.

3.3.4 Wikipedia - Liste der Staaten der Erde

I scraped the Wikipedia Liste der Staaten der Erde (*List of Sovereign States*) article and transformed it into a dictionary to translate the German country names of the EDB into English. The translation is a crucial step to match the countries in the EDB with the output of EpiTator to automatically create a labeled dataset (3.4). The List of Sovereign

```
def get_content_of_search_page(from_date, to_date, page_num):
  return (requests
          .get(f'https://www.promedmail.org/ajax/runSearch.php?'
               f'pagenum={page_num}&'
               f'search=&date1={from_date}&'
               f'date2={to_date}'
               )
          .content
          .decode('utf-8')
          )
```

Listing 3: The ProMED scraping core function. It executes a formatted Ajax GET request (indicated as a string in the `requests.get` method) for a certain date range and page number which returns a list of ProMED article URLs in the form of `'https://www.promedmail.org/direct.php?id=6400233'`. Everything in curly brackets is replaced by the function parameters.

States contains, besides others, the state's common, formal, and English name, and also the ISO-2 and ISO-3 abbreviation. The code for scraping the table from the Wikipedia page is shown in Lis. 4

```
# Request the HTML content from Wikipedia and parse it with BeautifulSoup
page = requests.get("https://de.wikipedia.org/wiki/Liste_der_Staaten_der_Erde")
soup = BeautifulSoup(page.content, "html.parser")

# Find table with all countries with HTML tag "table",
# the CSS class "wikitable sortable zebra", and the "tbody" tag
table_soup = soup.find("table", class_="wikitable sortable zebra")
body_soup = table_soup.find("tbody")

# Get entries of all countries form table
country_soup = body_soup.find_all("tr")
```

Listing 4: The Python code extract to scrape the Liste der Staaten der Erde table from Wikipedia using BeautifulSoup. The table is extracted using the `table, tbody` and `tr` tag and the `wikitable sortable zebra` class.

3.3.5 Wikidata and RKI-Internal Data

I queried all disease names in German and English from Wikidata (Lis. 5) and used them as a dictionary to transform all German disease names of the EDB to English to be able to match them with EpiTator's output to create a labeled dataset (3.4). Additionally, I used a list of RKI-internal abbreviation to translate them to their full English name.

```
endpoint_url = "https://query.wikidata.org/sparql"
query = """SELECT Distinct ?itemLabel_DE   ?itemLabel_EN WHERE {
            ?item wdt:P31 wd:Q12136.
            OPTIONAL{
            ?item rdfs:label ?itemLabel_DE.
            FILTER (lang(?itemLabel_DE) = "de"). }
            ?item rdfs:label ?itemLabel_EN.
            FILTER (lang(?itemLabel_EN) = "en").
            }"""
disease_translation_dictonary = get_results_of_sparql(endpoint_url, query)
```

Listing 5: The SPARQL request extract made to retrieve a list of tuples with the German and English disease name from Wikidata where `wdt:P31 wd:Q12136` is the item name of the disease list in Wikidata.

3.4 Preprocessing

The EDB is an unstructured Excel sheet which means that the data types were not restricted, no uniform vocabulary was used, and formatting errors occurred several times. After introducing the EDB, and the scraping of dictionaries to transform EDB entries into English, I show the necessary steps to clean the EDB, make it machine-readable, and transform it into a controlled English vocabulary to be comparable with the output of EpiTator for label creation.

3.4.1 Variables

The EDB has many entries where only a handful is mandatory. During the preprocessing, I only kept the entries that were mandatory to fill which are the **URL** of the article, the **disease**, the **country** of the outbreak, and the **confirmed case count** with the **date** (of the count).

The preprocessing steps were the removal of empty rows, trailing white spaces, and the split-up of several entries in one cell to several rows or the merge of redundant columns to be in accordance with the tidy data concept [Wickham, 2014]. The crucial part of the preprocessing was, however, the transformation of the variables into a controlled vocabulary.

3.4.2 Controlled Vocabulary

Dates The dates needed to be transformed into a pandas Timestamp object on which arithmetic operations and comparisons are possible. This was done with the `totimestamp()` method of pandas. If the method is provided with the information whether date or month is put first, `totimestamp()` automatically recognizes several forms of dates and transfers

them into the correct Timestamp object.

Counts Every case count entry in the EDB was cleaned from any non-digit entry. In the case of several numerical values, the first one was taken.

URLs I removed invalid URLs, and transformed ProMED URLs into a uniform style. The ProMED URLs in the EDB are written with both, `http` and `https` as a protocol. Furthermore, ProMED URLs in the EDB are written like '`https://www.promedmail.org/post/6400233`', as an example, or without `www.` preceding the URL. However, to be consistent with the scraped URL format returned by Lis. 3, I transformed all URLs into the form '`https://www.promedmail.org/direct.php?id=6400233`'.

Countries and diseases Country and disease names needed to be translated to make them comparable to the output of EpiTator. Therefore, I used the scraped dictionaries, as in 3.3.4 and 3.3.5, through which I eradicated spelling mistakes and translated these words to English. The procedure is shown in Algo. 1.

3.4.3 NLP-Pipeline

The WHO DONs and the ProMED Mail articles needed to be prepared for the ML algorithms. In the following, I describe the steps of the NLP-pipeline each text underwent before being fed into a classification algorithm.

3.4.3.1 Literal Processing

For all scraped websites, I replaced UTF-8 control characters with a single whitespace character illustrated by the following Python code: `string = "".join(char if unicodedata.category(char)[0] != "C" else ' ' for char in string)`.

3.4.3.2 Embedding

I used Flair to apply pretrained GloVe embeddings with 50 dimensions on each token of the dataset. GloVe has the advantage to cover global textual information better (2.2.8.2). The small size of the word embeddings was important to keep computational time low to increase the number of test iterations.

For an embedding comparison, I trained my own word embeddings using scraped ProMED and WHO DON articles. Since GloVe's global property was not necessary for the small corpus and due it high memory demand, I trained 50-dimensional embeddings using word2vec with the continuous skip-gram approach. This approach better captures rare words (2.2.8.1) which are typical for expert literature like epidemiological articles.

Input: $t_{\neg controlled}$
Result: $t_{controlled}$
`/* Initialize */`
dictionary : $K_{\neg controlled} \rightarrow V_{controlled}$;
if $t_{\neg controlled} \in K$ **then**
  return dictionary($t_{\neg controlled}$);
else

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

  `/* Where lev is the Levenshtein distance of word` a `and` b `at string` i `of` a `and` j `ob` b. `*/`
  $t_{corrected} \leftarrow \arg\min_{k \in K} \mathrm{lev}_{\neg controlled,k}$;
  if $t_{corrected} \in K$ **then**
    return dictionary($t_{corrected}$);
  else
    $t_{corrected} \leftarrow$ endsOrStartsWith($t_{\neg controlled}, dictionary$);
    return dictionary($t_{corrected}$);
  end
end

Algorithm 1: Translation algorithm to transform input to controlled vocabulary. If initially, no translation is available, the most similar written word is chosen using the Levenshtein distance $\mathrm{lev}_{a,b}(i,j)$. The word $k \in K$ with the shortest Levenshtein distance to the input word $t_{\neg controlled}$ is picked to correct the input word. If there is still no match to the controlled vocabulary, the algorithm assumes a shortened form and searches for an overlap in the first or last letters of the strings. Otherwise, no match is found.

3.5 Classification

I tested two methods for the key information extraction: First, taking the most frequent entity occurring in a text as the key entity (e.g., making the most common disease entity the primary thematized disease in the text) and second, train a classifier based on sentences that contain such entities. The training data were all sentences-tokenized ProMED and WHO DON articles of 2018. Sentences containing the keywords found in the corresponding EDB entry are the positive training labels. All other sentences containing keyword entities of the same category (e.g., confirmed cases) recognized by EpiTator, but are not found in the respective EDB entry, are counterexamples for the classifier.

Furthermore, I trained a classifier to be able to distinguish between *relevant* and *irrelevant* outbreak news based on the whole text, and based on the embeddings of the text.

The full pipeline of all steps necessary before the assembly of the labeled training dataset is depicted in Fig. 3.2

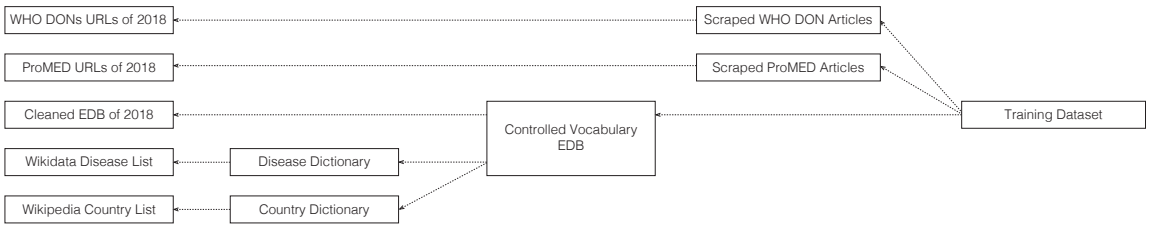

Figure 3.2: A depiction of all the dependencies during the assembly of the labeled training dataset.

3.5.1 Naive Keyword Extraction

During the reading of epidemiological news, I noticed that the key entities of a text are repeatedly mentioned. Therefore, I decided to naively determine a keyword entity by choosing the most frequent entity, i.e., the key disease entity would be the disease mentioned most frequently as shown in Lis. 6.

```
def return_most_occurring_entities(list_of_entities):
  list_of_entity_occurrence_tuple = [(key, len(list(group)))
                                     for key, group
                                     in groupby(sorted(list_of_strings))]
  most_occurring_string = max(list_of_entity_occurrence_tuple,
                              key=itemgetter(1))[0]
  return most_occurring_string
```

Listing 6: A simplified Python function to detect the most occurring entity in a list of entities.

3.5.2 Naive Bayes Classifier

I used the multinomial and complement NBC for the text classification using all WHO DONs and ProMED Mail articles of the year 2018. Articles denoted in the EDB are labeled *relevant* and the rest is labeled *irrelevant*. I applied tf-idf to balance term frequency discrepancies and removed stop words using NLTK's stop word list.

For the keyword extraction, I used the multinomial and Bernoulli NBC on labeled sentence-entity tuples. Sentences of texts that contain entities found in the corresponding EDB entity entry are labeled as *is key* whereas the rest is labeled *is not key*. I kept the stop words for the keyword extraction since I was expecting formulations such as *"112 confirmed cases as of 07.05.2018"*, where *"...as of..."* is an indicator for a key date (4.2.1).

3.5.3 Support Vector Machine

For the text classification, I used the average of all word embeddings as the document embedding with which I trained an SVM, using the radial basis function as the kernel. This kernel is better suited to classify not linearly separable classes. I chose a penalty parameter $C = 1$ to avoid overfitting to the imbalanced classes. Finally, I chose a kernel function coefficient of $\gamma = \frac{1}{50}$ which is a standard value based on the number of features, in this case, the length of the document embedding vector.

3.5.4 Logistic Regression

For the text classification, I used the average of all word embeddings as the document embedding with which I trained the logistic regression classifier. I applied L2 regularization with a regularization strength $C = 1$ to avoid overfitting to the imbalanced classes.

3.5.5 k-Nearest Neighbor

For the text classification, I used the average of all word embeddings as the document embedding with which I trained a k-nearest neighbor classifier with $k = 5$ to balance sensitivity to noise, as for small k. Furthermore, I used the Euclidian distance to measure distances and did not weight the distances.

3.5.6 Deep Learning

During the text classification, I used the average of all word embeddings as the document embedding. First, I fed it into a multilayer perceptron with 100 neurons in the hidden layer, a rectified linear unit as the activation function, the Adam optimizer using the default values during training, and L2 penalty to avoid overfitting. Also, I used a CNN to iterate over all word embeddings using an automatically selected feature map size by SpaCy, and average pooling. For the classifier trained on embeddings, I used ADASYN to upsample the minority (*relevant*) class of the dataset.

3.6 Web App

Using Flask and DataTables, I built a web app that allows the entry of an URL where then the text is extracted, summarized, and put into a database which can be downloaded in various formats. The web app also contains a method to start an automatic scraping of all articles from a specified source up to the last analyzed article from this source. These texts are then annotated and dumped into the database.

CHAPTER 4

Results and Evaluation

4.1 EDB Analysis

A substantial part of my work was the exploitation of the EDB and the scraped epidemiological news. In the following, I introduce how I recovered parts of the EDB for my further analyses and how I managed to scrape a large amount of epidemiological data with a limited time profile.

4.1.1 Source Determination

Before I could train a classifier to detect relevant articles or extract meaningful key information from texts, I needed to create a labeled dataset from the EDB. Therefore, I needed to collect all articles that were read as part of INIG's surveillance routine. However, writing scripts for IR (such as scraping) can be time-consuming, so I needed to narrow down the, at that moment, 75 sources used in the EDB.

The decision which source to extract was then made in regard to the relevance and the complexity to retrieve information from this source. First, I extracted all URLs from the EDB and clustered them based on their netloc (a first level domain such as www.rki.de) to rank the URLs according to their frequency (Fig. 4.1) to then only focus on the most used sources. Then, I pinpointed those sources that are the most accessible. To do so, I evaluated INIG's reading checklist which includes all sources that are mandatory to observe. In this evaluation, I determined the data type, conducted exemplary information extractions to assess the accessibility, and evaluated the relevance of these sources by their articles (Tab. 4.1). Generally, information extraction from PDFs and Emails was more difficult. PDFs, unlike HTML, do not consist of a markup language where specific information can be individually extracted. Emails, on the other hand, were difficult to access due to privacy reasons and the information in them was usually less structured.

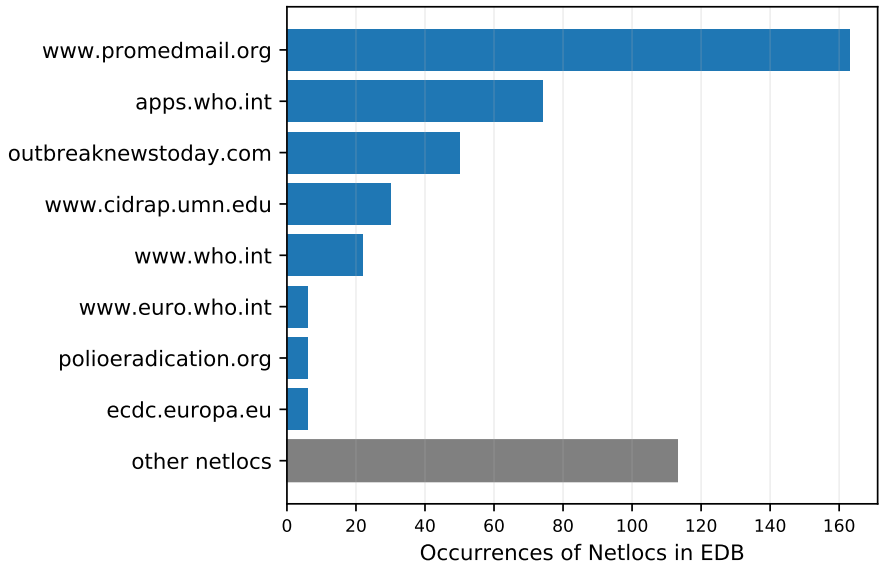

Figure 4.1: The netloc frequency of the most used sources of the EDB. Shown in grey is the sum of EDB entries referencing the other 67 netlocs not shown in this figure.

Table 4.1: An evaluation of INIG's reading checklist by source, data quality, and accessibility. The data format refers to the final data format of the epidemiological text. The data quality describes whether a source only contains information relevant for epidemiological surveillance or also research findings and ongoing projects (mixed content). The difficulty evaluation is based on exemplary IR from these sources where *easy* posed no difficulty, *intermediate* would have required additional work but was promising to function and *hard* was unsure whether it could work satisfactorily.

Source	Data Format	Data Quality	Accessibility
CIDRAP	HTML	Mixed content	Intermediate
ProMED Mail	HTML	Only relevant	Easy
WHO DONs	HTML	Only relevant	Easy
EIOS daily digest	Email	Only relevant	Hard
OutbreakNewsToday	HTML	Mixed content	Intermediate
ECDC Report	Email	Only relevant	Hard
WHO Afro Bulletin	PDF	Only relevant	Hard
EuroSurveillance	PDF and HTML	Mixed content	Intermediate
WHO WER	PDF	Mixed content	Hard
ECDC CDTR	PDF	Only relevant	Intermediate
WHO EMRO	PDF	Mixed content	Hard
WHP PAHO	PDF	Only relevant	Intermediate

The final decision was to then to scrape ProMED Mail article (163 entries in the EDB), and WHO DONs (22 entries in the EDB) indicated by `www.who.int` in Fig. 4.1. Both are the most used HTML sources of the EDB that are also easy to scrape and only contain relevant information. Cidrap and OutbreakNewsToday, although being quoted more often

than WHO DONs, also publish non-epidemiological articles and are harder to scrape wherefore I disregarded them. In sum, articles from 185 EDB entries (of 557) were scraped for the assembly of a labeled dataset using only two scrapers. Note, `apps.who.int` in Fig. 4.1, also part of WHO, is not a WHO DON but includes several epidemiological bulletins that are published as PDFs.

4.1.2 Data Quality

Due to the unrestrained column settings, every entry in the EDB was free text with spelling mistakes, inconsistent formatting, or left empty. The transferral of the EDB into a controlled vocabulary, as described in 3.4.2, was a vital step since it made more data points useable. Tab. 4.2 shows the performance of the data preparation.

Table 4.2: A performance measure of the transmission of the EDB to a controlled vocabulary. The table shows the number of valid entries before and after the preprocessing and the number of empty columns per keyword. The numbers refer to the whole EDB with 557 entries. The numbers in parentheses refer to the training dataset with 155 entries quoting ProMED or WHO DON articles.

Keyword	Valid Before Preproc.	Valid After Preproc.	Invalid After Preproc.	Empty Before and After Preproc.
Date	168 (37)	229 (141)	19 (7)	309 (7)
Case count	299 (87)	394 (105)	18 (0)	145 (50)
Country	355 (15)	494 (148)	17 (4)	46 (3)
Disease	231 (0)	332 (111)	16 (16)	209 (28)

The transferral to the controlled vocabulary retrieved around 100 keywords per keyword column for the full EDB. Typical invalid entries for dates were dates formulated as free text like *"the first two weeks in May"*. Another common mistake the entering of symptoms instead of diseases. A typical invalid country entry contained a city or region name instead of the country. Of 185 EDB entries referencing ProMED and WHO DON articles, 155 used valid URLs. Of the 30 invalid URLs, 8 were ProMED URLs without an article id but just (`"www.promedmail.org"`). The reason why only the first level domain was entered so often is the dynamical loading of articles in ProMED. When a user reads different articles on ProMED, the URL does not change. The person who wants to enter a ProMED article into the EDB needs to open the printable version of the article in order to see the unique URL of this article. The other URLs were correctly written but referred to articles that were not available anymore which happens when articles are withdrawn.

Nevertheless, the preprocessing was able to increase the number of usable data points. For the datasets for key information extraction, 141 EDB entries with a valid date could be retrieved and 105 entries with a valid count. For the testing of the naive keyword

selection (Lis. 6) I additionally retrieved 148 entries with valid country names and 111 entries with a valid disease name.

4.2 Key Information Extraction

Although EpiTator extracted only relevant entity classes, they still needed to be filtered for the key entity of this class, i.e., the one that would need to be put into the EDB. To do so, I used the most frequent entity of an entity class. For example, when Epitator finds relevant disease names the most frequent one is chosen (Lis. 6).

First, I used the naive key information extraction as a baseline. This worked well for the disease and country key information extraction but not for the date and count extraction. Thus, I trained a multinomial and Bernoulli NBC using all sentences of a text containing a specific entity class. The label *is key* was given to those sentences where the extracted entity matched the entity class found in the EDB for this text and *is not key* to the others.

4.2.1 Results

Except for one occurrence of a disease that was not recognized by EpiTator, all countries and diseases were detected correctly using the naive key information extraction. However, this approach performed poorly for the key information extraction of count and date entities. Only 12 count entities of 105 were correctly retrieved (recall of 0.11 for the *is key* class and IBA of 0.10) following the naive approach and no date entity out of 141 (recall of 0.00 for the *is key* class and IBA of 0.00).

The performance of both classifier for the key information extraction of count entities is shown in Tab. 4.3 and for date entities in Tab. 4.4.

Regarding the recall of the *is key* class and the IBA score, the Bernoulli NBC was performing better with a recall of 0.24 and an IBA of 0.23 for the count key information extraction. The multinomial NBC had a value of 0.00 for both these measures, and the naive approach had a recall of 0.11 and an IBA of 0.10. The Bernoulli NBC was also better performing than the multinomial NBC for the date key information extraction with a recall of 0.11 and an IBA 0.10 while the multinomial NBC and the naive approach had a value of 0.00 for both these measures.

The corresponding ROC curves and their AUC values are shown in Fig. 4.2. The AUC for the count extraction was higher (0.72) for the Bernoulli NBC than the multinomial NBC (0.50). The AUC for the date extraction was 0.83 for the multinomial NBC and 0.48 for the Bernoulli NBC.

Since there were strong expectations which phrases indicated the mentioning of a key entity such as *"... confirmed cases..."* or *"... cases ... as of..."*, I retrieved those tokens

Table 4.3: The performance evaluation of the count key information extraction. For each classifier and label, the precision (Pre.), recall (Rec.), specificity (Spec.), F1, index balanced accuracy (IBA) with $\alpha = 0.1$, and support (Sup.) is given. The recall of the *is key* class was the main objective of the classifier, and since the dataset was imbalanced, the IBA was a better measure for the overall accuracy. Blue values are the best in both category and orange values the worst.

	Pre.	Rec.	Spec.	F1	IBA	Sup
Multinomial Naive Bayes						
is not key	0.91	1.00	0.00	0.95	0.00	446
is key	0.00	0.00	1.00	0.00	0.00	43
Average/Total	0.83	0.91	0.09	0.87	0.00	489
Bernoulli Naive Bayes						
is not key	0.93	0.90	0.24	0.91	0.23	447
is key	0.18	0.24	0.90	0.21	0.20	42
Average/Total	0.86	0.84	0.29	0.85	0.23	489

Table 4.4: The performance evaluation of the date key information extraction. For each classifier and label, the precision (Pre.), recall (Rec.), specificity (Spec.), F1, index balanced accuracy (IBA) with $\alpha = 0.1$, and support (Sup.) is given. The recall of the *is key* class was the main objective of the classifier, and since the dataset was imbalanced, the IBA was a better measure for the overall accuracy. Blue values are the best in both category and orange values the worst.

	Pre.	Rec.	Spec.	F1	IBA	Sup
Multinomial Naive Bayes						
is not key	0.67	1.00	0.00	0.80	0.00	26
is key	0.00	0.00	1.00	0.00	0.00	27
Average/Total	0.44	0.67	0.33	0.53	0.00	39
Bernoulli Naive Bayes						
is not key	0.78	0.93	0.11	0.85	0.11	30
is key	0.33	0.11	0.93	0.17	0.10	9
Average/Total	0.68	0.74	0.30	0.69	0.11	39

that were strong indicators for the *is key* label (Tab. 4.5).

4.2.2 Evaluation

It is interesting that the multinomial NBC was by a large margin better performing in the date entity selection from sentences than the Bernoulli NBC according to the AUC (Fig. 4.2). This finding contradicts the former expectation that Bernoulli NBC is typically preferred in short text classifications. However, the contrary applies to the count entity recognition where the Bernoulli NBC indeed is better, yet with a smaller difference. When

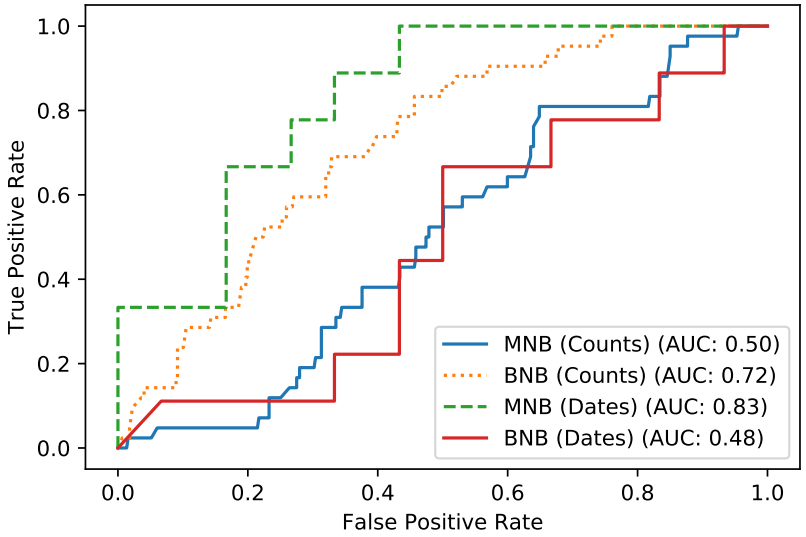

Figure 4.2: A ROC-curve comparison of the different key information extraction classifier trained using a Multinomial naive Bayes (MNB) and Bernoulli naive Bayes (BNB) classifier with the respective value for the AUC.

Table 4.5: The most important words during the classification of the multinomial NBC to detect the key entities Counts and Dates.

Entity Class	Word	Positive (%)
Counts	variant	31.1
	poultry	27.1
	Laibin	27.1
	42-year-old	22.2
	strains.	19.2
	province.Aug	19.2
	13For	19.2
Dates	worm	6.0
	occurring	5.3
	Nothern	5.3
	emerging	5.3
	patients	4.5
	South	4.1
	deaths	3.9

we regard the recall for the *is key* class and the average IBA score, then for the date and count key information extraction, the Bernoulli NBC was the better performing algorithm (Tab. 4.3 and 4.4). In my opinion, the recall for the *relevant* class together with the IBA is critical since only correct classifications will decrease the burden for epidemiologists to enter key information into a database while the IBA gives a good measure about the overall performance of the model. The AUC does not seem to resemble this desired property

because the multinomial NBC with the poor recall appears to be superior to the Bernoulli NBC for the date key information extraction according to the AUC. Thus, to maximize the usability of the key information extraction, I would prefer using the Bernoulli NBC for date and count extraction. It is noteworthy that both Bernoulli NBCs were better than the naive approach despite their small training data. For training the count entity extraction, 2445 labeled sentences (198 *is key* sentences) were used and for training the date entity extraction 195 labeled sentences (46 *is key* sentences). Surely, with more and better quality data that would result from a structured database, better results can be expected.

Also, the words most responsible for a positive classification for an important count or date entity (Tab. 4.5) did not match the expectation for words like *confirmed* or phrases like *as of*. Especially the count classification is based on poor tokenization (*13For*, *strains.*, and *province.Aug*). The detection of such unique (wrong) tokens in positive examples indicates a severe overfit of the classifier. The extracted words look more reasonable for the date classification, and the word *patients* or *death* appear to make sense. Both words could be used in a sentence where the author would also mention the date of the confirmed case numbers. This result suggests that the data needs to undergo better preprocessing to avoid false tokenization to achieve better results with less overfit. Note, while Tab. 4.5 displays important features from the multinomial NBC, there is no large difference to be expected to the features of the Bernoulli NBC. Especially, because the Bernoulli NBC tends to overfit even more. The overall performance of the date and count key information extraction, however, is not good enough to be used in production. The disease and country key information extraction, on the other hand, works already well enough.

4.3 Article Recommendation

For the detection of relevant disease outbreak articles, I used the scraped WHO DONs and ProMED Mail articles together with the EDB entries, all of the year 2018, to build a labeled dataset which consists of 155 *relevant* and 3077 *irrelevant* texts. Since I intended to also use word embeddings besides the bag-of-words approach to training a classifier, I visualized the word embeddings of the training data pretrained on the Wikipedia and Gigaword Corpus and self-trained embeddings using WHO DON and ProMED articles. Finally, I compared different classifier trained with the tf-idf transformed bag-of-word approach and document embeddings balanced with ADASYN.

4.3.1 Results

The pretrained embeddings showed the typical clusters of similar tokens such as digits at the coordinates (-40, -90), names (-75, 25), adjectives (50, -75), and medical terms at (0,

25) (Fig. 4.3). The self-trained embeddings appear to cover more medical terms from the center up to the upper right corner that also seem to be clustering. However, it lacks other typical clusters (Fig. 4.3). Also, the sizable medical term cluster is not well concentrated.

Since the primary goal of the trained classifier is the recognition of relevant articles, the recall for the *relevant* label is essential. Tab. 4.6 shows that the SVM has the highest score for recall of relevant articles (0.79). Also, the IBA, which takes the class imbalance into account opposed to the F1 score, is the highest on average for the support vector classifier (0.47). The CNN is the worst performing classifier for both these measures (0.00 for both). The complement naive Bayes and multinomial naive Bayes classifier performed equally good (recall of 0.10 and IBA of 0.11).

For comparison, I also plotted the ROC-curves and AUC values of all classifier (Fig. 4.4). The AUC value for the CNN is the highest (0.86) and for both NBC the worst (0.46).

4.3.2 Evaluation

The pretrained word embeddings do not cover the same amount of technical vocabulary compared to the self-trained embeddings. On the other hand, the pretrained embeddings have other semantically meaningful clusters like adjectives or names which are missing in the self-trained embeddings. It is likely that a more extensive corpus of epidemiological articles would show prominent technical vocabulary cluster and common word clusters at the same time which then would yield a better foundation for training classifiers.

Against expectations (Tab. 2.1), the multinomial and complement NBC had an identical performance (Tab. 4.6 and Fig. 4.4) although the complement NBC tackles problems occurring in imbalanced datasets specifically. Also, the deep learning methods worked worse concerning the imbalanced class measures and were only excelling regarding common criteria like the F1 score or the AUC value. It appears that the AUC value, although commonly used to measure the performance of classifier trained with imbalanced classes, does not give a good measure for the classification of *relevant* articles as the IBA or recall does since the CNN has the highest AUC score but has a recall of 0.00 for *relevant* article (Tab. 4.6). A reason for the poor performance of the CNN is that it overfitted. Overfitting can be avoided with dropout (random removal of nodes in the network during training time to minimize highly specified nodes), regularization (e.g., L2 to punish strong weighting of nodes), and early stopping (to minimize the difference of losses between the test and validation set). I believe that with the aforementioned adjustments, the CNN could perform better. However, building a well-adjusted model was not in the scope of this thesis.

For now, the support vector machine is preferred due to its good IBA and recall value for the *relevant* class. Although the relevance classification has not a strong performance, it could already aid epidemiologists. The model could be retrained every time articles are

Table 4.6: The performance evaluation of the relevance classification. For each classifier and label, the precision (Pre.), recall (Rec.), specificity (Spec.), F1, index balanced accuracy (IBA) with $\alpha = 0.1$, and support (Sup.) is given. The support vector classifier uses the radial basis function (RBF) as a kernel. Blue values are the best in their category and orange values the worst.

	Pre.	Rec.	Spec.	F1	IBA	Sup
Multinomial Naive Bayes						
Irrelevant	0.96	0.98	0.10	0.97	0.11	617
Relevant	0.21	0.10	0.98	0.14	0.09	30
Average/Total	0.92	0.94	0.14	0.93	0.11	647
Complement Naive Bayes						
Irrelevant	0.96	0.98	0,10	0.97	0.11	617
Relevant	0.21	0.10	0.98	0.14	0.09	30
Average/Total	0.92	0.94	0.14	0.93	0.11	647
Logistic Regression						
Irrelevant	0.97	0.67	0.63	0.79	0.42	762
Relevant	0.09	0.63	0.67	0.15	0.42	38
Average/Total	0.93	0.66	0.63	0.76	0.42	800
k-Nearest Neighbor Classifier						
Irrelevant	0.97	0.77	0.53	0.86	0.41	762
Relevant	0.10	0.53	0.77	0.17	0.39	38
Average/Total	0.93	0.75	0.54	0.82	0.41	800
Support Vector Machine (RBF)						
Irrelevant	0.98	0.60	0.79	0.74	0.46	762
Relevant	0.09	0.79	0.60	0.16	0.48	38
Average/Total	0.94	0.61	0.78	0.72	0.47	800
Multilayer Perceptron						
Irrelevant	0.97	0.78	0.58	0.87	0.46	762
Relevant	0.12	0.58	0.78	0.19	0.44	38
Average/Total	0.93	0.77	0.59	0.83	0.46	800
Convolutional Neural Network						
Irrelevant	0.95	1.00	0.00	0.98	0.00	762
Relevant	0.00	0.00	1.00	0.00	0.00	38
Average/Total	0.91	0.95	0.05	0.93	0.00	800

entered into the EDB to increase performance continuously. Until then the relevance score could just be displayed instead of using it to filter content.

4.4 Web App

Finally, I built a web application named "Aussinator" using Flask and Datatables to visualize a possible workflow of the keyword extraction and relevance scoring.

The web app allows the user to enter an URL for evaluation. The output of this evaluation is the extraction of key information of this article and a relevance score (Fig. 4.5). The app also allows a more automated workflow. The buttons `Get ProMED articles` or `Get WHO DONs` will trigger an automatic evaluation of all articles of the respective domain since the last review. Instead of building an automated evaluation update, this function allows the user to observe the functionality of the application better. Furthermore, the application has its own database that can be downloaded in different formats. That was an essential step at this time point since INIG was not sure how to proceed to store data.

Aussinator offers easy access to the information extraction and relevance scoring developed during this thesis. Furthermore, it provides enough freedom to the user during operation by providing a possibility to validate the output of the app. However, the live extraction is rather slow. The processing of one URL takes up to two seconds. This delay is particularly noticeable when extracting several articles as intended by `Get ProMED articles`. The preprocessing by EpiTator is the bottleneck. NER often is a slow process, and EpiTator looks up several entities which causes the delay. A solution to this would be a daemon process that regularly checks whether the sources of interest published new articles. If so, it would then start scraping, evaluating, and then store the key information and relevance score in the background to increase the retrieval speed for the update functions `Get ProMED articles` and `Get WHO DONs`.

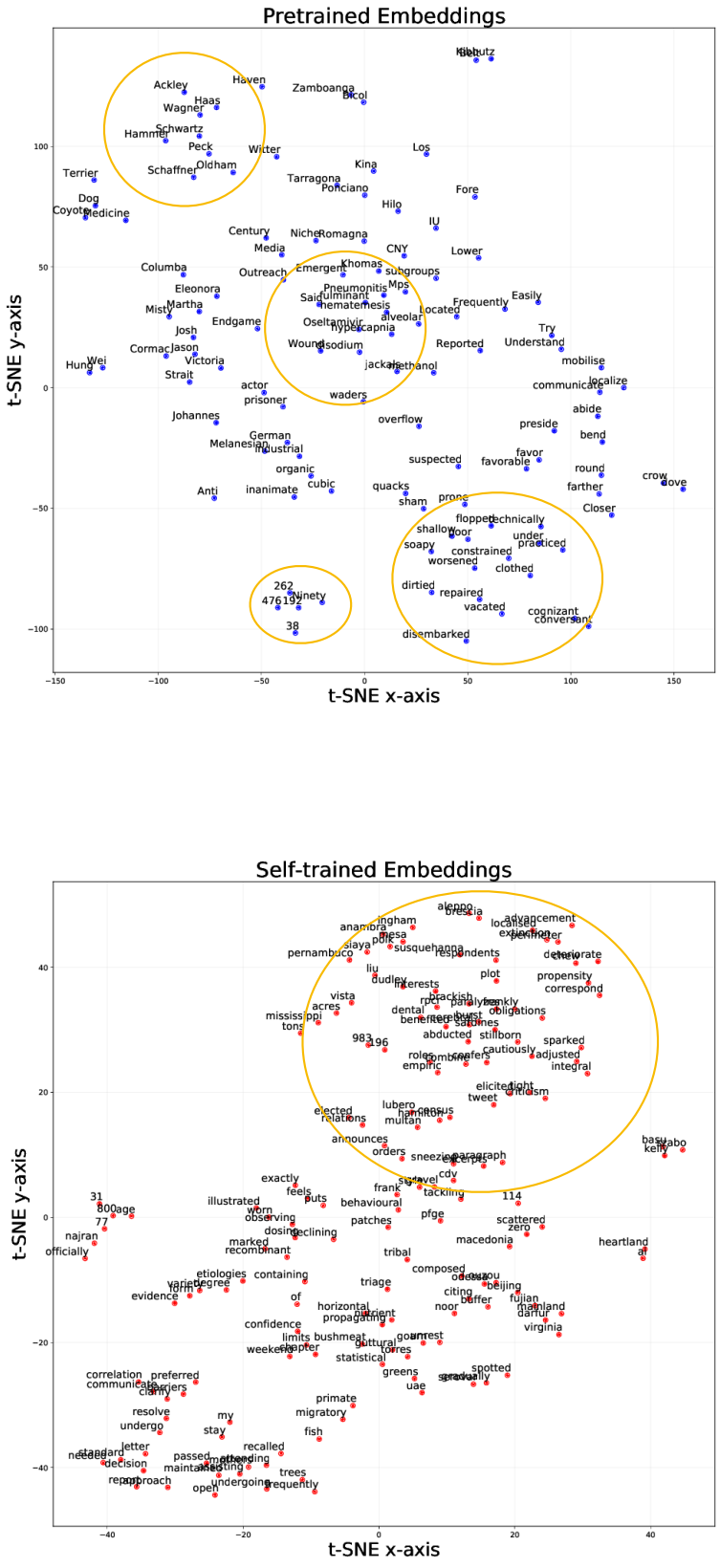

Figure 4.3: A comparison of word embeddings applied to the training data using embeddings pretrained on the Wikipedia and Gigaword corpus (blue) and on WHO DON and ProMED articles (red) using t-SNE for dimensionality reduction. The pretrained embeddings show cluster of digits (-40, -90), names (-75, 25), adjectives (50, -75), and medical terms at (0, 25) encircled in orange. The self-trained embeddings show a neighborhood of medical terms.

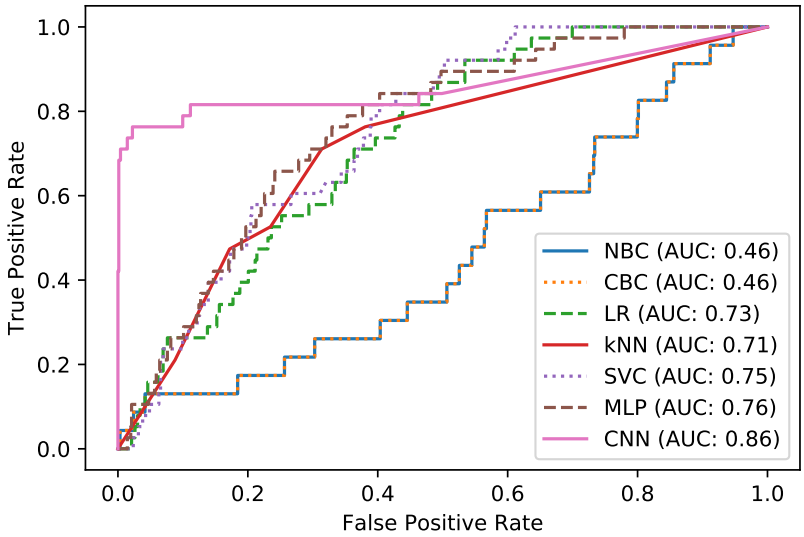

Figure 4.4: A ROC-curve comparison of the different relevance classifier trained using a Multinomial naive Bayes (MNB), complement naive Bayes (CNB) classifier, logistic regression (LR), k-nearest neighbor classifier(kNN), support vector machine (SVM), multilayer perceptron (MLP) and convolutional neural network (CNN) with the respective value for the AUC. Note, the curves of the MNB and CNB overlay.

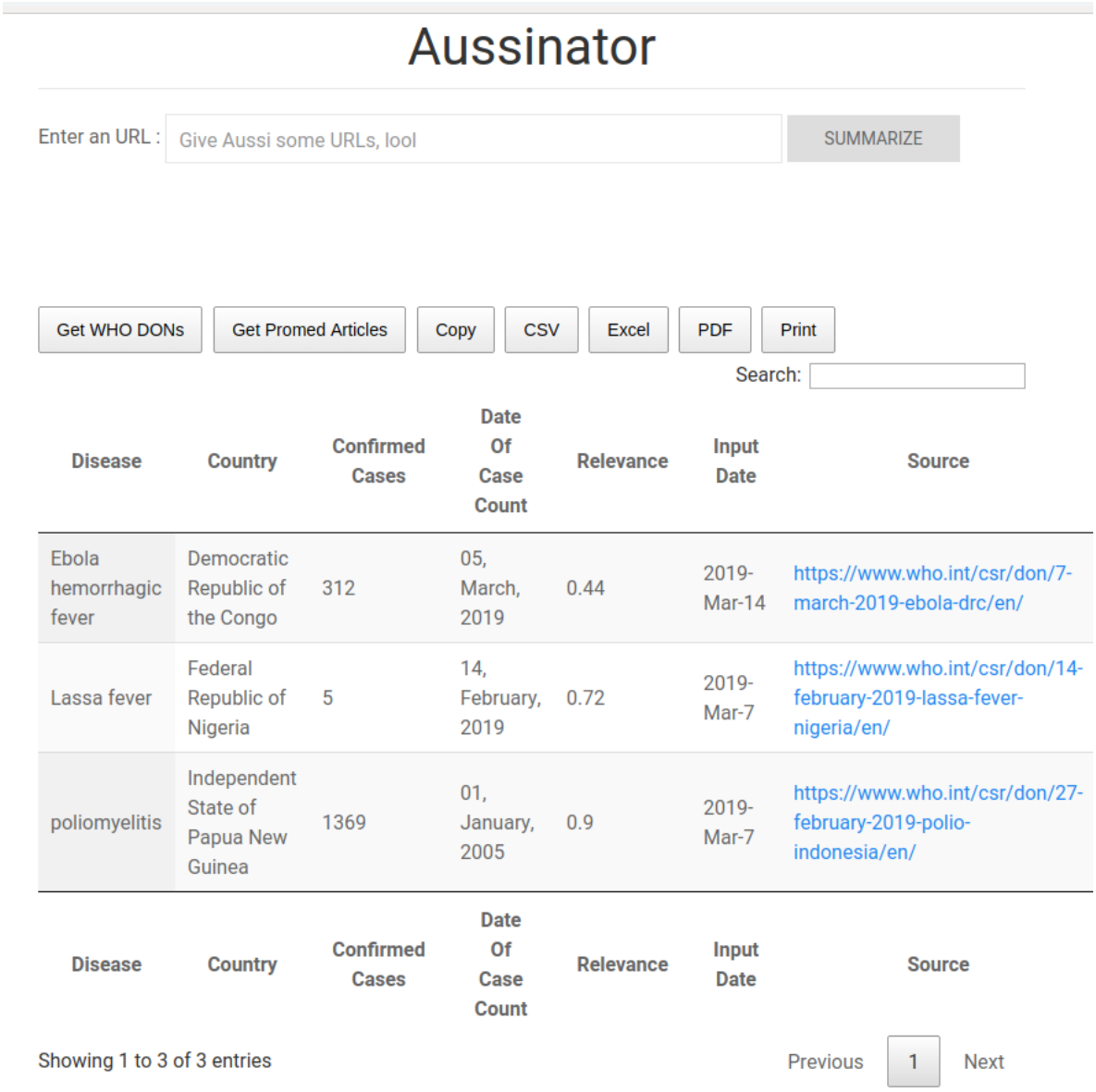

Figure 4.5: Aussinator, a Flask web application using the keyword extraction and relevance scoring trained as part of this thesis.

CHAPTER 5

Conclusion

The main objective was to show that it is feasible to utilize already available resources at the RKI, in this case, the Incident Database, using novel NLP methodology to improve epidemiological surveillance. Thereby, the outcome of this thesis also proves that EBS, even with smaller resources, is possible. By providing a web service and using open source libraries, I developed an scalable tool. This is of interest for the RKI which shares expertise with other countries and institutes.

Of course, more work is necessary to bring the Aussinator into production. The performance of the keyword extraction of the date and the count needs to be improved for better user experience. However, this point can be tackled as mentioned in the evaluation of the classifier (4.2.2). This is true for also other goals not met during this thesis such as the availability of the Aussinator in several languages and the avoidance and the uncovering of possible human biases in the evaluation of text relevance. It is possible to provide better classifications that work for different languages using multilingual word embeddings [Chen and Cardie, 2018], or a better keyword extraction using contextual embeddings [Devlin et al., 2018; Peters et al., 2018] which adjust the embedding based on the textual context. Primarily, the poor performing keyword extraction strongly depends on the local properties of the keywords to be extracted. Which entity is relevant most often depends on the words nearby and thus contextually adapted embeddings might increase the performance of the keyword extraction.

Also, tackling biases and personal preferences is essential to continue this project and make it save to use. It will be essential to show how the decisions of Aussinator are made, to win the approval of epidemiologists in the proceeding digitization of EBS. Thus, the relevance score needs to be made explainable by revealing *why* these decisions were made. Therefore, it would be desirable to visualize the text fragments on which the relevance classifier based its decision on, and which would also allow detecting biases in the classifier [Arras et al., 2017].

Since Aussinator is a web service, the next step would be to give several people access to this tool, and leverage the increased usage to train a base neural network for the classification and then use transfer learning, to adopt the network to individual preferences.

Finally, I only showed that it is possible to automate the keyword extraction and relevance scoring using the incident database. To, however, achieve production-ready results, it would be necessary to apply a separate study to select and finetune classification algorithms to reach better performances.

Bibliography

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. "What is relevant in a text document?": An interpretable machine learning approach. In *PloS one*, 2017.

V. Balakrishnan and E. Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. *Lecture Notes on Software Engineering*, pages 262–267, 2014. doi: 10.7763/LNSE.2014.V2.134. URL `http://eprints.um.edu.my/13423/`.

Yehoshua Bar-Hillel. Some Linguistic Problems Connected with Machine Translation. *Philosophy of Science*, 20(3):217–225, jul 1953. ISSN 0031-8248. doi: 10.1086/287266. URL `https://www.journals.uchicago.edu/doi/10.1086/287266`.

Yehoshua Bar-Hillel. The Present Status of Automatic Translation of Languages. *Advances in Computers*, 1:91–163, jan 1960. ISSN 0065-2458. doi: 10.1016/S0065-2458(08)60607-5. URL `https://www.sciencedirect.com/science/article/pii/S0065245808606075`.

Benjaming Bengfort, Tony Ojeda, and Rebecca Bilbro. *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*. O'Reilly Media, Inc., 2018. ISBN 9781491963043.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003. ISSN ISSN 1533-7928. URL `http://www.jmlr.org/papers/v3/bengio03a.html`.

Justus Benzler, Göran Kirchner, Michaela Diercke, and Andreas Gilsdorf. Das Projekt DEMIS. In *Der Hygieneinspektor*. Robert Koch-Institut, Infektionsepidemiologie, 2014. doi: 10.25646/1932. URL `https://edoc.rki.de/bitstream/handle/176904/2007/20teFClzrKjE.pdf?sequence=1{&}isAllowed=y`.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495, 9780596516499.

Bundesminesterium der Justiz und für Verbraucherschutz. Gesetz zur Verhütung und Bekämpfung von Infektionskrankheiten beim Menschen, 2001. URL https://www.gesetze-im-internet.de/ifsg/.

Xilun Chen and Claire Cardie. Unsupervised Multilingual Word Embeddings. aug 2018. URL http://arxiv.org/abs/1808.08933.

Code Google. Google Code Archive - Long-term storage for Google Code Project Hosting., 2013. URL https://code.google.com/archive/p/word2vec/.

Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. oct 2018. URL http://arxiv.org/abs/1810.04805.

Daniel Faensen, Hermann Claus, Justus Benzler, Andrea Ammon, Thomas Pfoch, T Breuer, and Gérard Krause. SurvNet@RKI - A multistate electronic reporting system for communicable diseases. In *EuroSurveillance*. Robert Koch-Institut, 2006. doi: http://dx.doi.org/10.25646/602.

Hermann Gröhe. Together today for a healthy tomorrow-Germany's role in global health. *Lancet (London, England)*, 390(10097):831–832, aug 2017. ISSN 1474-547X. doi: 10.1016/S0140-6736(17)31617-3. URL http://www.ncbi.nlm.nih.gov/pubmed/28684023.

Haibo He, Yang Bai, Garcia Edwardo A., and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, jun 2008. ISBN 978-1-4244-1820-6. doi: 10.1109/IJCNN.2008.4633969. URL http://ieeexplore.ieee.org/document/4633969/.

Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. jan 2018. ISSN 23330384. doi: arXiv:1801.06146v3. URL http://arxiv.org/abs/1801.06146.

Paul S. Jacobs, George R. Krupka, and Lisa F. Rau. Lexico-semantic pattern matching as a companion to parsing in text understanding. In *Proceedings of the workshop on Speech and Natural Language - HLT '91*, pages 337–341, Morristown, NJ, USA, 1991. Association for Computational Linguistics. doi: 10.3115/112405.112477. URL http://portal.acm.org/citation.cfm?doid=112405.112477.

Antonio Jimeno, Ernesto Jimenez-Ruiz, Vivian Lee, Sylvain Gaudan, Rafael Berlanga, and Dietrich Rebholz-Schuhmann. Assessment of disease named entity recognition on a corpus of annotated sentences. *BMC Bioinformatics*, 9(3):S3, apr 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-S3-S3. URL https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-S3-S3.

Tibor Kiss and Jan Strunk. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*, 32(4):485–525, dec 2006. ISSN 0891-2017. doi: 10.1162/coli.2006.32.4.485. URL http://www.mitpressjournals.org/doi/10.1162/coli.2006.32.4.485.

S C Kleene. Representation of Events in Nerve Nets and Finite Automata, 1951. URL https://apps.dtic.mil/docs/citations/ADA596138.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*, page 441, New York, New York, USA, 2010. ACM Press. ISBN 9781605588896. doi: 10.1145/1718487.1718542. URL http://portal.acm.org/citation.cfm?doid=1718487.1718542.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Charles X Ling and Victor S Sheng. Cost-Sensitive Learning and the Class Imbalance Problem. 2008.

Chundi Liu, Shunan Zhao, and Maksims Volkovs. Unsupervised Document Embedding With CNNs. 2018.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text Classification using String Kernels. *Journal of Machine Learning Research*, 2 (Feb):419–444, 2002. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v2/lodhi02a.html.

H. P. Luhn. Key word-in-context index for technical literature (kwic index). *American Documentation*, 11(4):288–295, oct 1960. ISSN 0096946X. doi: 10.1002/asi.5090110403. URL http://doi.wiley.com/10.1002/asi.5090110403.

Elliott Macklovitch and Michel Simard. Transsearch: A free translation memory on the world wide web. *Second International Conference On Language Resources and Evaluation, LREC2000*, 2000. URL http://rali.iro.umontreal.ca/rali/sites/default/files/publis/TS3MT.pdf.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008. ISBN 9780511809071. doi: 10.1017/CBO9780511809071. URL http://ebooks.cambridge.org/ref/id/CBO9780511809071.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313—-330, 1993. URL https://aclanthology.info/papers/J93-2004/j93-2004.

Lluís Màrquez and Horacio Rodríguez. Part-of-speech tagging using decision trees. In *Machine Learning: ECML-98*, pages 25–36. 1998. doi: 10.1007/BFb0026668. URL http://link.springer.com/10.1007/BFb0026668.

Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. *AAAI-98 workshop on learning for text categorization*, 752(1):41—-48, 1998. URL https://www.semanticscholar.org/paper/A-Comparison-of-Event-Models-for-Naive-Bayes-Text-McCallum-Nigam/04ce064505b1635583fa0d9cc07cac7e9ea993cc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN 0070428077. URL http://www.cs.cmu.edu/{~}tom/mlbook.html.

Oliver Mohr, Edward Velasco, Gerhard Fell, Florian Burckhardt, Gabriele Poggensee, and Tim Eckmanns. Die telefonische infektionsepidemiologische Bund-Länder-Lagekonferenz in Deutschland - Eine Zwischenbilanz nach drei Quartalen 2009. In *Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz*. Robert Koch-Institut, Infektionsepidemiologie, 2010. doi: 10.1007/s00103-010-1122-z. URL https://edoc.rki.de/bitstream/handle/176904/916/25RYrvOTxkGp6.pdf?sequence=1{&}isAllowed=y.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. Joint Lemmatization and Morphological Tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1272. URL http://aclweb.org/anthology/D15-1272.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticæ Investigationes*, 30(1):3–26, aug 2009. ISSN 0378-4169. doi: 10.1075/li.30.1.03nad. URL https://benjamins.com/catalog/li.30.1.03nad.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. feb 2018. URL http://arxiv.org/abs/1802.05365.

RANKS. Stopwords, 2019. URL https://www.ranks.nl/stopwords.

Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, Washington, DC, USA, 2003. AAAI Press. URL http://citeseerx.ist.psu.edu/viewdoc/citations?doi=10.1.1.13.8572.

Irina Rish. An Empirical Study of the Naïve Bayes Classifier. *IJCAI 2001 Work Empir Methods Artif Intell*, 3, 2001.

Robert Koch Institute. RKI - Institut, 2018. URL https://www.rki.de/DE/Content/Institut/institut{_}node.html.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009. ISBN 0136042597, 9780136042594.

Simon Tong and Daphne Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, 2(Nov):45–66, 2001. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v2/tong01a.html.

Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in NLP. In *Proceedings of the 14th conference on Computational linguistics* -, volume 4, pages 1106–1110, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. doi:

10.3115/992424.992434. URL http://portal.acm.org/citation.cfm?doid=992424.992434.

Qikang Wei, Tao Chen, Ruifeng Xu, Yulan He, and Lin Gui. Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks. *Database*, 62(4):727–737, oct 2011. ISSN 1758-0463. doi: 10.1093/database/baw140. URL https://academic.oup.com/database/article-lookup/doi/10.1093/database/baw140.

WHO. Early detection, assessment and response to acute public health events. *WHO*, 2014a. URL https://apps.who.int/iris/handle/10665/112667.

WHO. Epidemiology. *WHO*, 2014b. URL https://www.who.int/topics/epidemiology/en/.

WHO. Epidemic intelligence - systematic event detection, 2015. URL https://www.who.int/csr/alertresponse/epidemicintelligence/en/.

Hadley Wickham. Tidy Data. *Journal of Statistical Software*, 59(10):1–23, sep 2014. ISSN 1548-7660. doi: 10.18637/jss.v059.i10. URL http://www.jstatsoft.org/v59/i10/.

Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock. Word Mover's Embedding: From Word2Vec to Document Embedding. *CoRR*, abs/1811.0, 2018. URL http://arxiv.org/abs/1811.01713.

Vikas Yadav and Steven Bethard. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA, 2018. Association for Computational Linguistics. URL https://aclanthology.info/papers/C18-1182/c18-1182.

Index