TECHNICAL UNIVERSITY OF MUNICH

Department of Mathematics

TUM Data Innovation Lab

AI for Document Understanding

Authors:	Alican Ay, Wassim Boubaker, Oliver Kobsik, Cingis Alexander Kratochvil
Mentors:	M.Sc. Matthias Wissel, M.Sc. Arndt Kirchhoff, M.Sc. Matthias Rebs Capgemini Deutschland GmbH
Project lead:	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor:	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Abstract

Automating business processes is an important investment of many companies to simplify the workflow and accelerate frequent processes. Many companies have large amounts of documents, often scanned, which are used in the daily business. For instance, consider the case where an employee has to find specific information about a client out of thousands of scanned documents. In the average case, the employee has to work through hundreds of documents, which is very time-consuming.

To provide an automated solution, we worked on a pipeline that is able to classify documents into predetermined classes, recognize text from the scanned documents, extract keywords from the documents and serve as a platform that allows the end-user to search through the documents.

The methods we used to implement such a pipeline include document classification, where we used transfer learning on convolutional neural networks (CNNs) followed by a meta-classifier (a multilayer perceptron), serving as a stacked generalization scheme. To extract text from the scanned documents, we used optical character recognition (OCR) with Tesseract-OCR. Furthermore, we used named entity recognition (NER) to tag specific keywords in these documents, deploying CNNs, bi-directional long short-term memory networks (BiLSTMs) and conditional random fields (CRFs). These keywords belong to predefined classes such as organizations, locations, people, etc.

On top of that, we deployed a search platform using Solr whose results are visualized on a Banana dashboard. The whole pipeline is provided as a cloud-based web application using Flask monitored by OpenNebula.

Contents

1 Introduction

We live in the era of digitization. It has struck most fields (Automotive, Banking etc.) and corporate services (Recruiting, Management etc.). However, there still are many open opportunities that could accelerate corporate processes and thereby generate benefits and lower costs.
Within our project we decided, together with Capgemini, to tackle the digitization of paper-based processes, i.e. archiving in the public sector, experiment reports in R&D, medication prescriptions in healthcare etc.
The need for such a service on the German market has been validated and even mentioned lately (January 2019) by the *Süddeutsche Zeitung* in [52] (only available in German). The article depicts the struggle and the amount of work needed to bring order into the archive of two municipalities in Munich, Bavaria. Taking the space constraint and the physical work needed to access information into consideration, the digitization of the archive (or most of it) presents itself as a very fruitful solution, where unlike now, nothing has to be thrown away and sorting and retrieving information can be done through a computer program on a remote server.
Together with Capgemini, we designed and implemented an end-to-end solution to tackle the archiving problem within corporations and institutions.

Solution Architecture

The designed pipeline was proposed by Capgemini and is shown in Figure 1. The solution is composed of different micro-services that can either be called independently or within the complete pipeline.
The first micro-service is the document classification component that is handled in Chapter 2. It takes a scanned document as input and predicts its class among a finite predefined set of classes (e.g. emails, letters, advertisement etc.).
The second pipeline component consists of the Optical Character Recognition (Chapter 3), which takes scanned documents as input and extracts the text within these images.

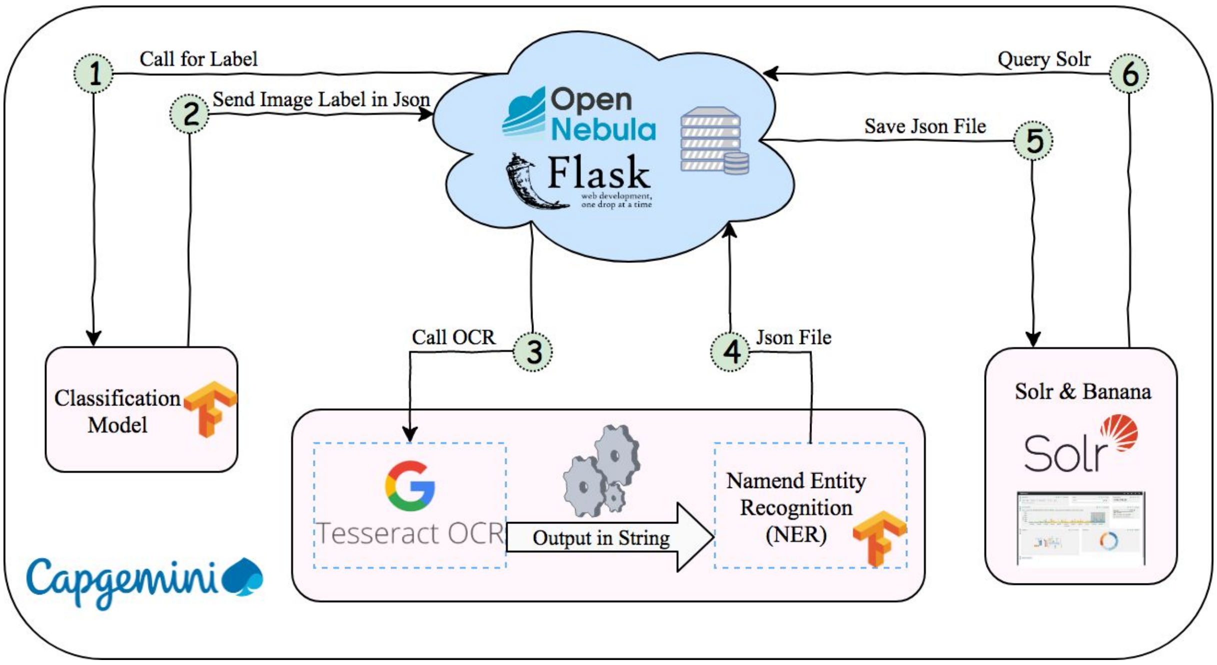


Figure 1: Project Pipeline [28], [27], [29]

The third micro-service provided by our project is the Named Entity Recognition module that identifies certain entities in a given text, e.g. people, locations or organizations as explained in Chapter 4.
Finally, the product is made usable through the Banana User Interface (UI). A Solr search-server runs in the back-end and organizes the output of our service pipeline. Banana comes on top to display and make our processed data searchable and beneficial for our user.
With the assistance of Capgemini, the project pipeline was built as a web application using Flask and runs fully on separate virtual machines on the LRZ cloud using OpenNebula. The cloud infrastructure and the communication between the different micro-services is depicted in Chapter 6.
In the conclusion, we wrap-up the whole pipeline and go over some potential extensions for the final product.

Project Management & Planning

We applied for this project as a team and we decided to work as a team. Therefore, we chose to follow the Scrum [41] method; an agile software development methodology.

Team Roles

Under the Scrum method, we had the following structure:

- The product owner: Capgemini.
- A Scrum master: One of our team members controlling the work progress, plans and issues.
- 4 developers: The four of us developing the final product.

Used Frameworks

We based our communication on Slack, since it offers a broad team collaboration and organization services such as work channels, archiving, reminders etc.
The second organization tool we used for our project management was Gitlab. Apart from using it as a code collaboration platform, Gitlab offers an outstanding backlog tracking board where we created, revisited and assigned tasks, together with Capgemini.

Working Methodology

The Scrum method implies frequent and regular backlog feedback and discussion with the product owner, Capgemini. Therefore, we met on a weekly basis to monitor the progress of our project, resolve issues and plan for the next meeting.
We also organized weekly development task forces, where we, the 4 developers, worked on our weekly tasks, exchanged information and defined assignments for the next week. One month before the project submission, we started having daily Scrum calls to fight frustration and quickly resolve issues.

Gitlab Repository

The complete implementation is to be found on the following Gitlab repository https://gitlab.lrz.de/ga65red/capgai-doc/.
The master branch contains some README files and some manuals while the code has been completely deployed on the dev branch.

2 Document Classification

One of the first steps in document understanding is to recognize what type of document we are looking at. There are many possibilities to tackle this. We decided to apply a combination of procedures presented in [16] by Harley, Ufkes and Derpanis and in [9] by Das, Roy, Bhattacharya, Parui. Both papers propose an image-based method, i.e. classifying a document without using its textual content.
Figure 2 depicts the classification pipeline step by step.

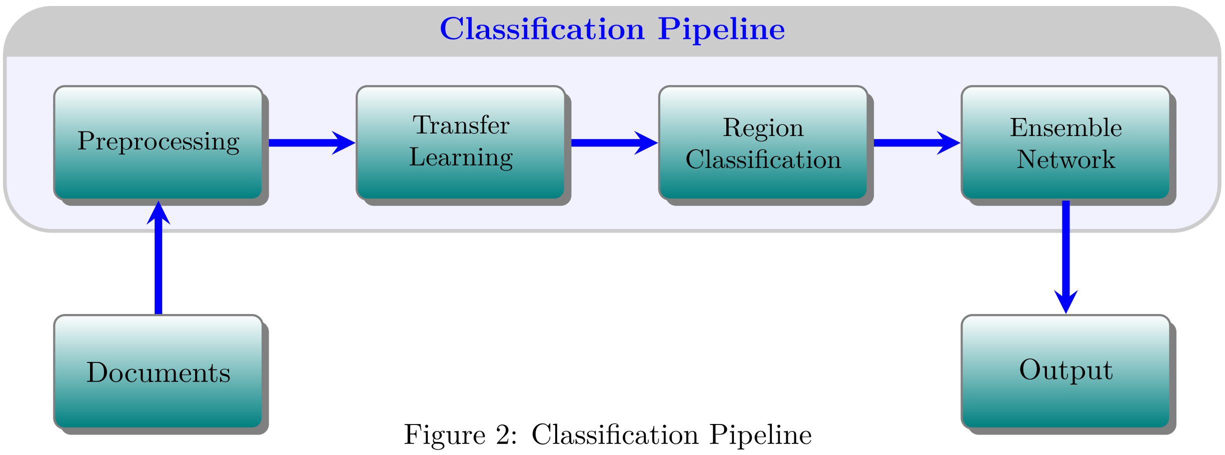
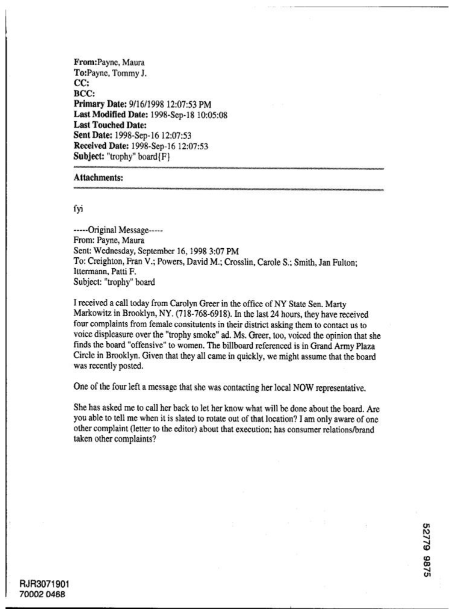


Figure 2: Classification Pipeline

2.1 Problem Definition

Given an email or a scanned letter the task is to recognize the type of the document. Specifically, the task is to implement a classifier that is trained, validated and tested on the *Ryerson Vision Lab Complex Document Information Processing* (RVL-CDIP) dataset [16] using the programming language *Python 3.6* and the machine learning library *Tensorflow* [1] .
In Figures 3 and 4, two samples of the email and letter classes from the RVL-CDIP dataset are shown.
One of the challenges in image-based document classification is the *intra-class variability* [16], i.e. visual appearance of documents within a same class can strongly vary. Another challenge is the *inter-class similarity* [16], i.e. the visual appearance of documents between different classes can be very similar. For instance, Figure 4 shows the intra-class variability since both documents belong to the letters class. However, both are not visually similar. On the other hand, both emails shown in Figure 3 are visually similar to the letter in Figure 4a.
Upon request from Capgemini, we restricted our work to email and letter classes from beginning on. These two classes are widely used formats in day-to-day business. In particular, these two classes are well-suited for our use-case because the documents are scans of digital texts. That means that OCR will perform better than other classes which contain also hand-written documents. Moreover, we guarantee that the upcoming NER micro-service obtains better quality data. Lastly, due to intra-class variability and inter-class similarity in emails and letters, we were interested in how well our proposed solution is going to perform.



From: Payne, Maura
To: Payne, Tommy J.
CC:
BCC:
Primary Date: 9/16/1998 12:07:53 PM
Last Modified Date: 1998-Sep-18 10:05:08
Last Touched Date:
Sent Date: 1998-Sep-16 12:07:53
Received Date: 1998-Sep-16 12:07:53
Subject: "trophy" board{F}

Attachments:

fyi

-----Original Message-----
From: Payne, Maura
Sent: Wednesday, September 16, 1998 3:07 PM
To: Creighton, Fran V.; Powers, David M.; Crosslin, Carole S.; Smith, Jan Fulton; Ittermann, Patti F.
Subject: "trophy" board

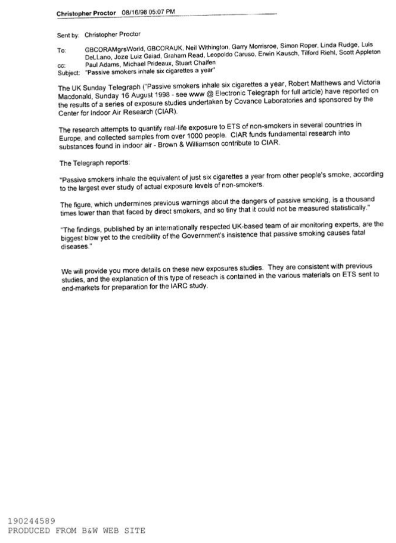
I received a call today from Carolyn Greer in the office of NY State Sen. Marty Markowitz in Brooklyn, NY. (718-768-6918). In the last 24 hours, they have received four complaints from female consitutents in their district asking them to contact us to voice displeasure over the "trophy smoke" ad. Ms. Greer, too, voiced the opinion that she finds the board "offensive" to women. The billboard referenced is in Grand Army Plaza Circle in Brooklyn. Given that they all came in quickly, we might assume that the board was recently posted.

One of the four left a message that she was contacting her local NOW representative.

She has asked me to call her back to let her know what will be done about the board. Are you able to tell me when it is slated to rotate out of that location? I am only aware of one other complaint (letter to the editor) about that execution; has consumer relations/brand taken other complaints?

52779 9875

RJR3071901
70002 0468



Christopher Proctor 08/16/98 05:07 PM

Sent by: Christopher Proctor

To: GBCORAMgrsWorld, GBCORAUK, Neil Withington, Garry Morrisroe, Simon Roper, Linda Rudge, Luis DelLano, Joze Luiz Gaiad, Graham Read, Leopoldo Caruso, Erwin Kausch, Tilford Riehl, Scott Appleton
cc: Paul Adams, Michael Prideaux, Stuart Chalfen
Subject: "Passive smokers inhale six cigarettes a year"

The UK Sunday Telegraph ("Passive smokers inhale six cigarettes a year, Robert Matthews and Victoria Macdonald, Sunday 16 August 1998 - see www @ Electronic Telegraph for full article) have reported on the results of a series of exposure studies undertaken by Covance Laboratories and sponsored by the Center for Indoor Air Research (CIAR).

The research attempts to quantify real-life exposure to ETS of non-smokers in several countries in Europe, and collected samples from over 1000 people. CIAR funds fundamental research into substances found in indoor air - Brown & Williamson contribute to CIAR.

The Telegraph reports:

"Passive smokers inhale the equivalent of just six cigarettes a year from other people's smoke, according to the largest ever study of actual exposure levels of non-smokers.

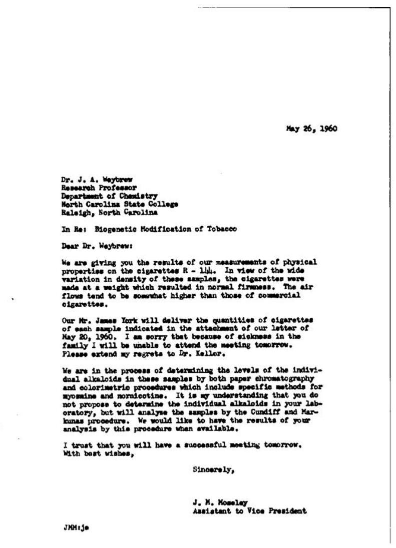
The figure, which undermines previous warnings about the dangers of passive smoking, is a thousand times lower than that faced by direct smokers, and so tiny that it could not be measured statistically."

"The findings, published by an internationally respected UK-based team of air monitoring experts, are the biggest blow yet to the credibility of the Government's insistence that passive smoking causes fatal diseases."

We will provide you more details on these new exposures studies. They are consistent with previous studies, and the explanation of this type of reseach is contained in the various materials on ETS sent to end-markets for preparation for the IARC study.

190244589
PRODUCED FROM B&W WEB SITE

Figure 3: Two samples of the email class taken from [16]



May 26, 1960

Dr. J. A. Weybrew
Research Professor
Department of Chemistry
North Carolina State College
Raleigh, North Carolina

In Re: Biogenetic Modification of Tobacco

Dear Dr. Weybrew:

We are giving you the results of our measurements of physical properties on the cigarettes R - 144. In view of the wide variation in density of these samples, the cigarettes were made at a weight which resulted in normal firmness. The air flows tend to be somewhat higher than those of commercial cigarettes.

Our Mr. James York will deliver the quantities of cigarettes of each sample indicated in the attachment of our letter of May 20, 1960. I am sorry that because of sickness in the family I will be unable to attend the meeting tomorrow. Please extend my regrets to Dr. Keller.

We are in the process of determining the levels of the individual alkaloids in these samples by both paper chromatography and colorimetric procedures which include specific methods for myosmine and nornicotine. It is my understanding that you do not propose to determine the individual alkaloids in your laboratory, but will analyse the samples by the Cundiff and Markunas procedure. We would like to have the results of your analysis by this procedure when available.

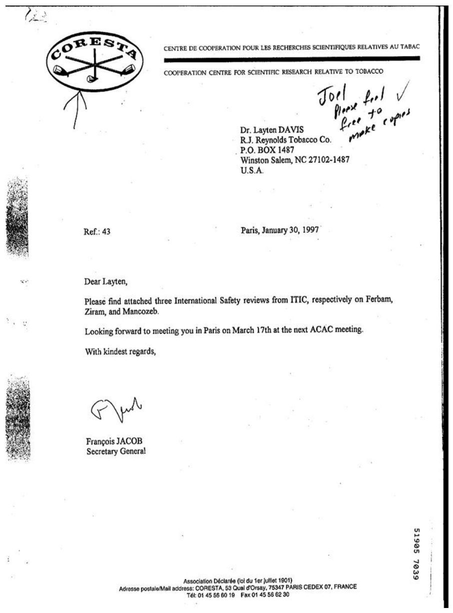
I trust that you will have a successful meeting tomorrow. With best wishes,

Sincerely,

J. M. Moseley
Assistant to Vice President

JMM:je

(a)



CORESTA

CENTRE DE COOPERATION POUR LES RECHERCHES SCIENTIFIQUES RELATIVES AU TABAC

COOPERATION CENTRE FOR SCIENTIFIC RESEARCH RELATIVE TO TOBACCO

Joel
Please feel ✓
free to
make copies

Dr. Layten DAVIS
R.J. Reynolds Tobacco Co.
P.O. BOX 1487
Winston Salem, NC 27102-1487
U.S.A.

Ref.: 43

Paris, January 30, 1997

Dear Layten,

Please find attached three International Safety reviews from ITIC, respectively on Ferbam, Ziram, and Mancozeb.

Looking forward to meeting you in Paris on March 17th at the next ACAC meeting.

With kindest regards,

[illegible]

François JACOB
Secretary General

51905 7039

Association Déclarée (loi du 1er juillet 1901)
Adresse postale/Mail address: CORESTA, 53 Quai d'Orsay, 75347 PARIS CEDEX 07, FRANCE
Tél: 01 45 56 60 19 Fax 01 45 56 62 30

(b)

Figure 4: Two samples of the letter class taken from [16]

2.2 First Steps

Once the problem had been defined and discussed with Capgemini, each team member looked up, read and presented a different paper regarding document classification. We went through the following papers: [16], [9], [4] and [10].
The paper [4] used handcrafted features from scanned documents which represent the documents as graphs. This meant that we would have to first label the RVL-CDIP dataset. Therefore, we decided not to implement the proposed model. The paper [10] proposes to apply a *Support*

Vector Machine (SVM) classifier on the content of the scanned documents, which implies that we would have to extract the text within the scanned documents. This would require a highly reliable OCR micro-service. At this phase of the project, we could not guarantee that. Thus, we decided not to implement the SVM classifier.
The papers [9] and [16] present classifiers that are image-based, trained on the RVL-CDIP dataset, and are using deep *convolutional neural networks* (CNNs). Since the topic of neural networks and deep learning matched one of our learning goals, we settled for these classifiers. Moreover, the implementation can be done conveniently using Tensorflow; one skill we wanted to add to our toolkit.

2.3 Dataset: RVL-CDIP

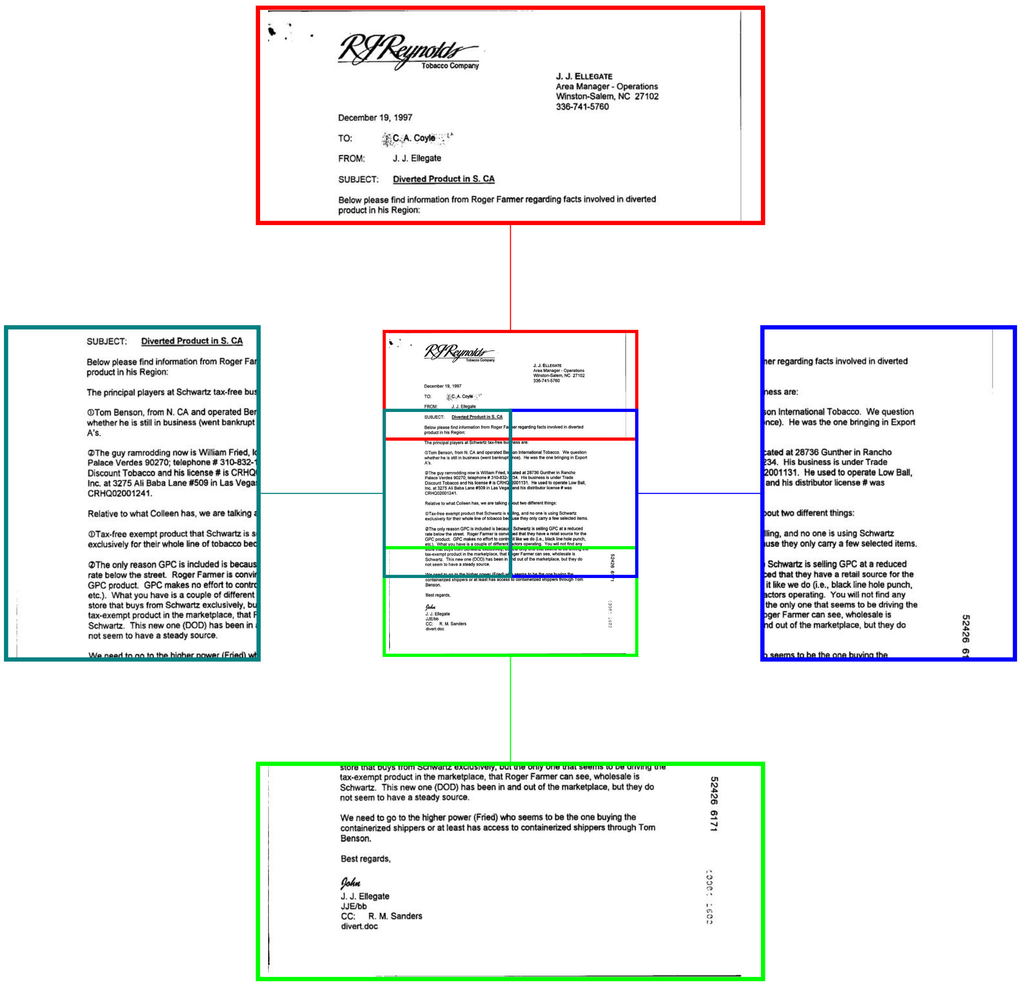
RVL-CDIP dataset is a collection of scanned documents from public records of lawsuits against American tobacco firms [16]. It offers in total 16 different document classes. Each class contains in total 25.000 scanned documents. As explained in Subsection 2.1, we used the classes emails and letters. The document classes are all balanced. This is an important property since it will prevent a possible bias toward a class of our trained classifier. Table 1 shows the balanced property for email and letter classes.

Table 1: Distribution of the email and letter classes

	Emails	Letters
Train	20010	20106
Validation	2485	2430
Test	2505	2464

2.4 Data Preprocessing

Firstly, the sizes of the scanned documents were unified. All scanned documents were resized to $780 \times 600 \times 3$, where the first two dimensions represent the height and width in pixels of the scanned documents while the last dimension represents the *red-green-blue* (RGB)-Channels. Since our data samples are grayscale images, duplicating and stacking them yielded an RGB representation, as done in [9]. Further, as shown in Figure 5, we extracted from each document the header, bottom, left and right regions.
The last step was to resize each document to $224 \times 224 \times 3$. This was necessary to get the classifier working as we will point out later.



RJReynolds Tobacco Company

J. J. ELLEGATE
Area Manager - Operations
Winston-Salem, NC 27102
336-741-5760

December 19, 1997

TO: C. A. Coyle

FROM: J. J. Ellegate

SUBJECT: Diverted Product in S. CA

Below please find information from Roger Farmer regarding facts involved in diverted product in his Region:

The principal players at Schwartz tax-free bus

①Tom Benson, from N. CA and operated Ber
whether he is still in business (went bankrupt
A's.

②The guy ramrodding now is William Fried, lo
Palace Verdes 90270; telephone # 310-832-1
Discount Tobacco and his license # is CRHQ
Inc. at 3275 Ali Baba Lane #509 in Las Vegas
CRHQ02001241.

Relative to what Colleen has, we are talking a

①Tax-free exempt product that Schwartz is s
exclusively for their whole line of tobacco bec

②The only reason GPC is included is becaus
rate below the street. Roger Farmer is convin
GPC product. GPC makes no effort to contro
etc.). What you have is a couple of different
store that buys from Schwartz exclusively, bu
tax-exempt product in the marketplace, that R
Schwartz. This new one (DOD) has been in a
not seem to have a steady source.

ner regarding facts involved in diverted

ness are:

son International Tobacco. We question
nce). He was the one bringing in Export

cated at 28736 Gunther in Rancho
234. His business is under Trade
2001131. He used to operate Low Ball,
and his distributor license # was

bout two different things:

lling, and no one is using Schwartz
use they only carry a few selected items.

Schwartz is selling GPC at a reduced
ced that they have a retail source for the
it like we do (i.e., black line hole punch,
actors operating. You will not find any
the only one that seems to be driving the
oger Farmer can see, wholesale is
nd out of the marketplace, but they do

o seems to be the one buying the

store that buys from Schwartz exclusively, but the only one that seems to be driving the tax-exempt product in the marketplace, that Roger Farmer can see, wholesale is Schwartz. This new one (DOD) has been in and out of the marketplace, but they do not seem to have a steady source.

We need to go to the higher power (Fried) who seems to be the one buying the containerized shippers or at least has access to containerized shippers through Tom Benson.

Best regards,

John
J. J. Ellegate
JJE/bb
CC: R. M. Sanders
divert.doc

52426 6171

Figure 5: Region based CNN

2.5 Model Architecture

Our implemented classifier is an ensemble network of *region-based* CNNs and a *holistic* CNN. The ensemble itself is a *feedforward neural network* (FFNN). In the following we describe in more details the CNNs, and the ensemble network.
Following [16], our model uses *transfer learning* [36] in order to accelerate the learning slope and reach a higher performance [50]. Both [16] and [9] use *inter-domain transfer learning* that consists in transferring learned features from a model trained on *ImageNet* [11] to our specific task. The latter dataset is a collection of images from different domains that go from animals to vehicles and plants etc., and therefore, very different from our training dataset. According to [15], transfer learning helps the model with generalization of slightly different data (e.g. letters or emails that do not originate from RVL-CDIP dataset).

2.5.1 Region-Based CNNs and Holistic CNN

The network architectures of the region-based CNNs and the holistic CNN are identical. However, the training data varies depending on the region.
The largest part of our CNNs represent the *convolutional part of VGG16* [45]. The weights were imported from a VGG16 image classifier trained on ImageNet [11]. Following [16] and [9] we

decided to freeze the imported weights. This decision supported also our agile development, since the convolutional part has 14,714,688 weight-parameters. Training this amount of parameters would be very time-consuming and would not allow fast feedback loops with Capgemini.
A trainable 256-unit *fully connected* (FC) layer comes on top of the non-trainable VGG16 block followed by a trainable 4096-unit FC layer. Both layers have a *rectified linear unit* (ReLU) as the activation function. We added a *dropout* [47] layer between them with dropout rate of 0.5 to reduce the risk of overfitting.
The output of the second FC layer is fed to the 2-dimensional output layer. The latter is equipped with a *softmax activation function* [18] to perform the final classification. The Figure 6 shows the architecture of our CNNs.

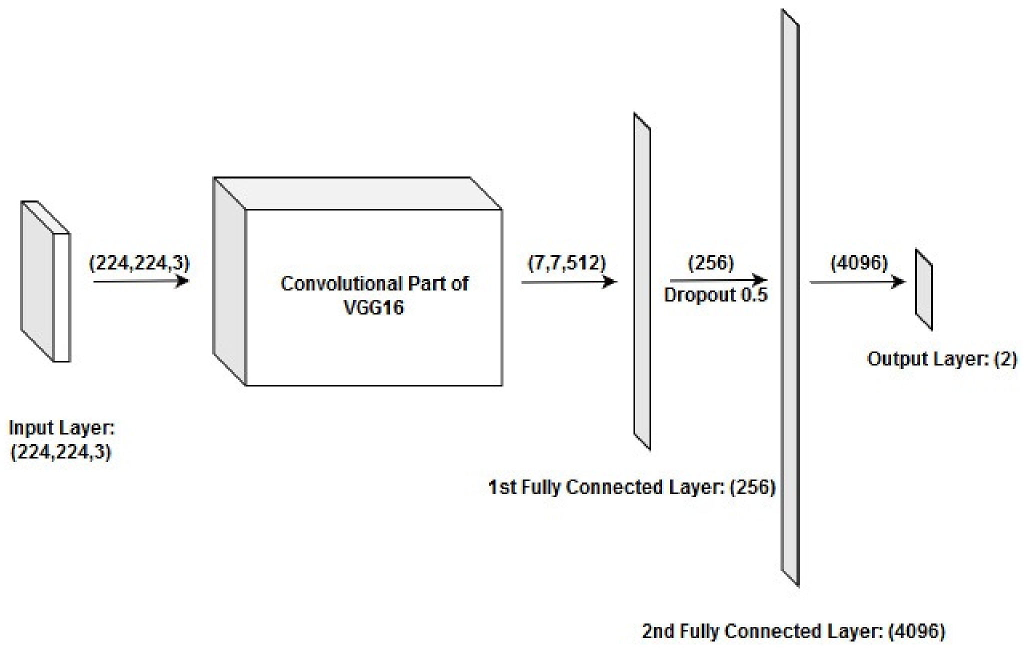

(224,224,3)
Convolutional Part of VGG16
(7,7,512)
(256)
Dropout 0.5
(4096)
Output Layer: (2)
Input Layer: (224,224,3)
1st Fully Connected Layer: (256)
2nd Fully Connected Layer: (4096)

Figure 6: Architecture of the region-based CNNs and the holistic CNNs

2.5.2 The Ensemble Network

Based on the paper [16], we decided that our ensemble network takes the first FC layers of all the CNNs. In this way, the ensemble network obtains all region-specific and holistic representations of the scanned document. These representations are concatenated as a 1280-dimensional vector and fed to an FC layer with a ReLU as activation function, as depicted in the Figure 7. Again, we use a dropout layer with a dropout rate of 0.75 right after the FC layer. The output layer is 2-dimensional with softmax activation function.
A. Das et al. [9] have compared several machine learning techniques (e.g. Ridge Regression, KNN, SVM, Extreme Learning Machines etc.) and it turned out that the *feedforward neural networks* (FFNNs) perform best in our use-case. These models belong to the stacked generalization schemes, first introduced in [54].

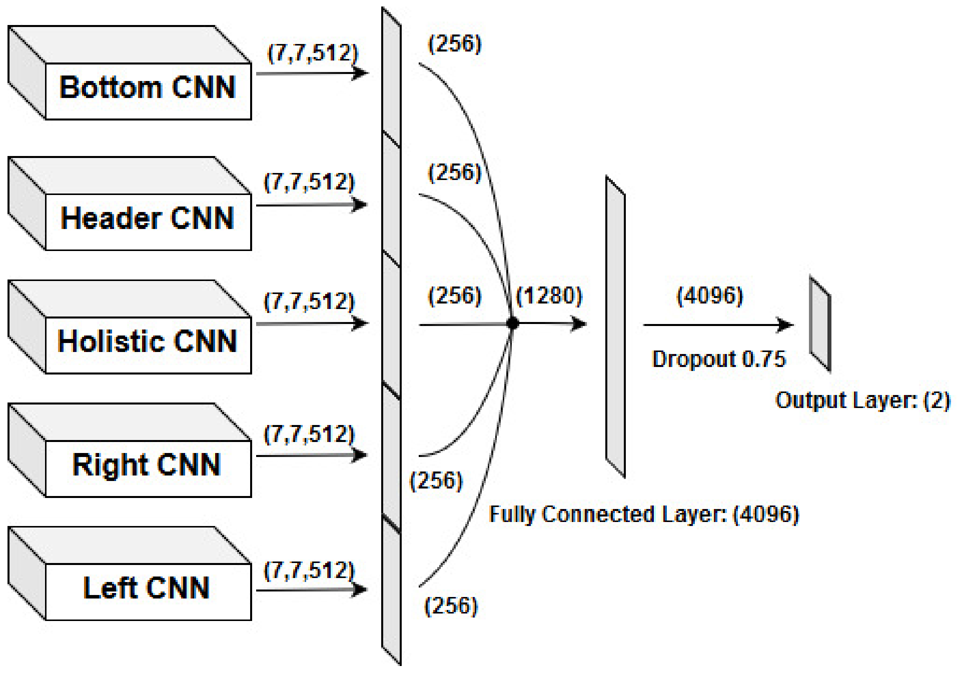


Figure 7: Architecture of the ensemble network

2.6 Implementation

We implemented the classifier in Python 3.6 and used the machine learning library Tensorflow (due to hardware limitations we used version 1.5 as explained later in Subsection 6.1.2) together with the high level API *Keras* [8]. For the data processing we used *NumPy* [35], the scientific computing package, and *Python Imaging Library* (PIL) [31]. The compatibility of Tensorflow with NumPy and PIL has shown to be very practical since all our data processing was done using the last two libraries. PIL offers a nice interface to switch between PIL image objects and NumPy arrays. The ensemble network was fully implemented using Keras.

2.7 Infrastructure for Training

Based on the one-day course *Introduction to the LRZ Supercomputing & Machine Learning Infrastructure*, we decided to use the LRZ *Linux Cluster* for training. Especially, the CooLMUC2 and CooLMUC3 Clusters [26]. To store the RVL-CDIP dataset, we used the LRZ *Input\Output server*. A cluster is similar to a CPU with a RAM and the Input\Output server is a Hard Disk Drive.
Since the LRZ Cluster is a shared cloud infrastructure, we had to use a *resource and workload manager* (SLURM) [25]. To train our classifier, we wrote bash scripts that can be found on our Gitlab repository, together with explanatory README files. For the bash scripts, we used special templates to communicate with SLURM. Once a bash script is submitted, SLURM decides when to run the (Python) program based on the resource requirements, the user's previous submissions and the current load on the LRZ Cluster.

2.8 Training

We trained our CNNs and ensemble network using one hyper-parameter configuration (default values from Keras) and have achieved satisfactory results. Therefore, we did not perform any hyper-parameter tuning or changes in our model architecture. As a loss function we used the *cross-entropy* [18]. For optimization we also used *Adam* [22], a stochastic gradient-based optimization algorithm. For Adam we used default parameters [20]. Lastly, we trained for 10 epochs, within each epoch there were 1252 iterations with a batch size of 32 scanned documents. We used the CooLMUC2 Cluster to train our model. We required 2 nodes, so 56 Intel Xeon CPUs and 128 GB of RAM. The training data was approximately 40.000 scanned documents, around 5.2 GB. The training runtime of a CNN classifier was approximately 9 hours and that of the ensemble network around 7 hours.

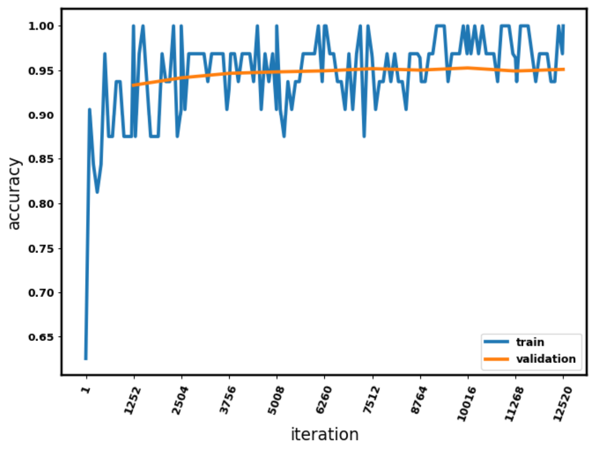
2.8.1 Accuracy of Region-Based CNNs and Holistic CNN

The region-based and holistic CNNs performed well on their own. We can see that in Table 2. The overperformance of the individual CNNs was predictable since the CNN classifiers presented in [16] reached accuracies of around 80% on the whole RVL-CDIP dataset, i.e. 16 classes unlike our binary classification problem. We should also note that these CNNs had a similarly large model architecture as our CNN classifier, but with a different CNN baseline than the VGG16. A quick look at Table 2 discards the fear of our models overfitting the training data, since they reach comparable results on both validation and test datasets.
Figure 8 shows that the right CNN reached its train and validation convergence accuracy within the first epoch, that is around 97% and 95%. The classifier converged in the remaining epochs. We observed the same training behaviour by the remaining CNNs. We can conclude from this behavior that our model complexity overtakes the problem we had to deal with. However, and also based on the work done in [16], this model should be able to tackle bigger classification tasks, depending on the real-life application intended by the product owner, Capgemini.

Table 2: Accuracies in the last epoch

	Header CNN	Bottom CNN	Right CNN	Left CNN	Holistic CNN
Train	0.9917	0.9876	0.9703	0.9764	0.9940
Validation	0.9782	0.9722	0.9524	0.9565	0.9861
Test	0.9777	0.9720	0.9447	0.9538	0.9855

Figure 9 shows that the right CNN already achieved its train and validation convergence accuracy within 300 from 1252 iterations and started oscillating between 90% and 97%. So, this might hint that there are just few features to be learnt that are sufficient to distinguish between an email and a letter in our dataset.



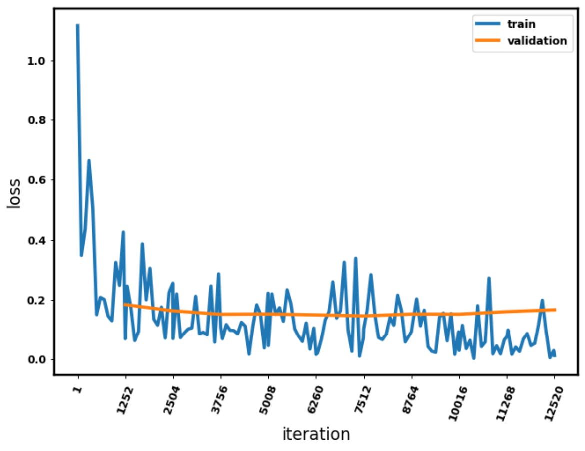


Figure 8: Accuracy (left) and loss function (right) plots by iteration of the right-based CNN. Every x-axis point corresponds to a training epoch (= 1252 iterations)

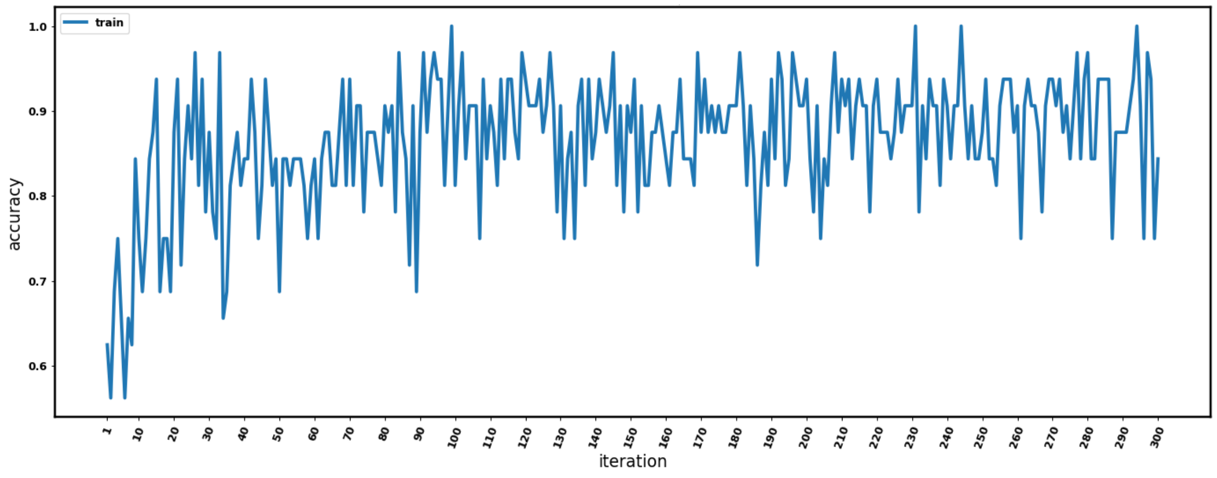


Figure 9: Accuracy plot of the first 300 iterations of the right-based CNN

2.8.2 Accuracy of the Ensemble Network

Since the ensemble network takes the output of the first FC layers of the region-based and holistic CNNs (that already perform well), we expected that it would reach a very comparable accuracy and hopefully a slightly better performance than the best CNN; in our case the holistic CNN (Table 2). According to [9], the ensemble network should reduce the generalization error compared to the input models.
Table 3 shows that the achieved accuracies indeed met our expectations. Figure 11 shows that the ensemble network achieves its convergence accuracy within only 30 iterations and oscillate around 99% on the training dataset after 300 iterations.
This was predictable since the input comes from the trained CNNs that achieved high accuracies. One can see from the Figure 10 that the ensemble network converged in the remaining epochs.

Table 3: Accuracies in the last epoch

	Ensemble
Train	0.9998
Validation	0.9877
Test	0.9893

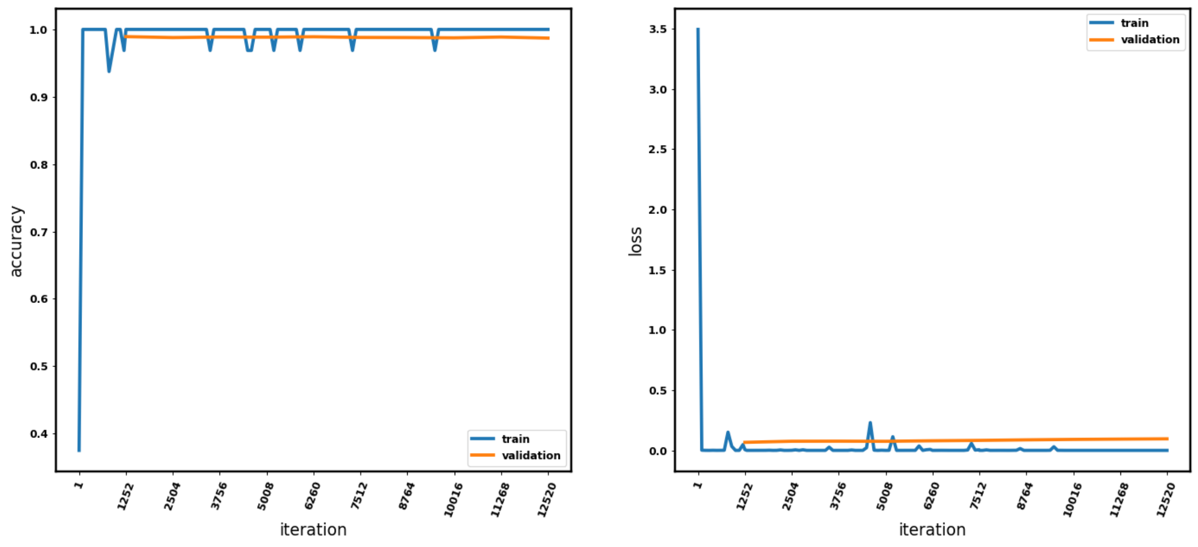


Figure 10: Accuracy plot of each 100th iteration and epoch of the ensemble network

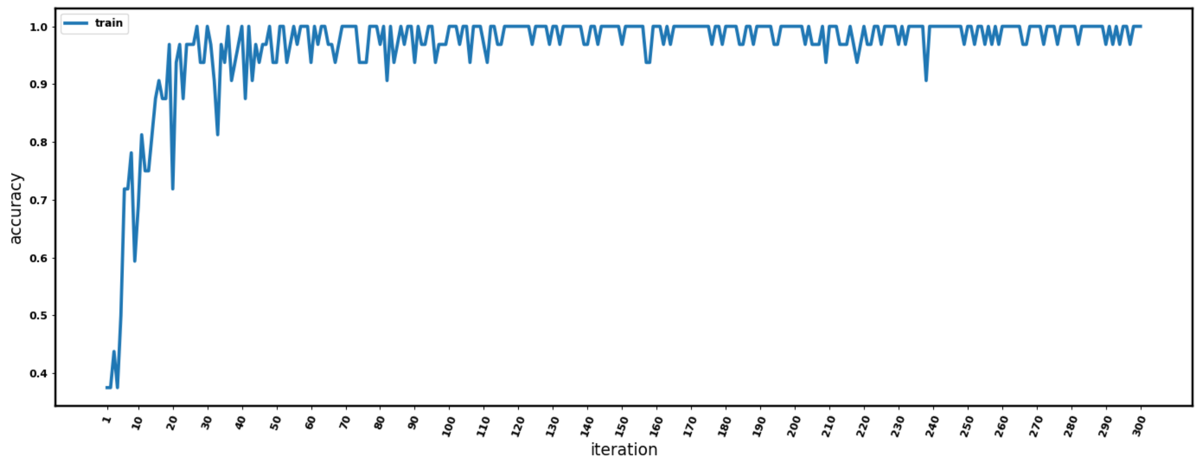


Figure 11: Accuracy plot of the first 300 iterations of the ensemble network

2.9 Troubleshooting

At the beginning of the semester, one of us was not able to submit bash scripts due to a certain account-related issue. This issue was resolved after contacting the LRZ Helpdesk. Moreover, at a later time, we decided to train our models once more to obtain better visualizations. At this time the LRZ Cluster was a bottleneck to one of our member since it was overloaded with other submissions and the member's account had low priority due to large amounts of previous submissions. The LRZ Helpdesk proposed to distribute the submissions among the other team members.

3 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a technique that recognizes text within digital images. We use it to find characters and their locations in scanned documents. Typical characters are letters, numbers and symbols. OCR can not only be used for printed text documents, but also for handwritten documents. Here, we consider two different OCR software algorithms. On the one hand, the open source software *Tesseract-OCR* [46], which runs locally. On the other hand, *OCR.Space* [34], which runs fully on a remote server and whose free API is limited in the number and size of processed images.
Figure 12 shows the main components discussed in the following subsections starting with the input being a scanned document.

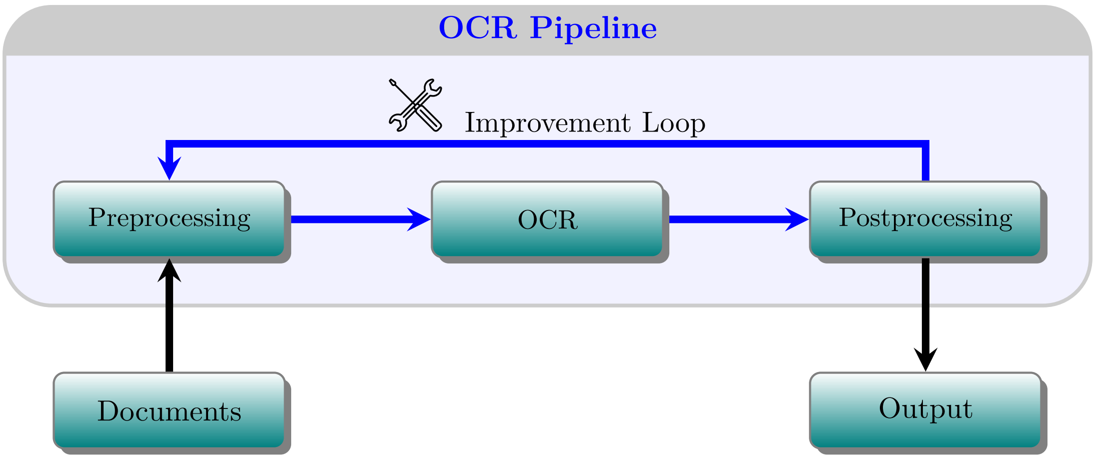


Figure 12: OCR Overview

3.1 Problem Definition

Our aim is to extract characters from scanned documents. Sometimes characters are too small, blurred or simply unreadable. Therefore, the used software should be able to detect the coherence of words or even sentences. The number of paper-based documents is enormous and present in almost every business area. Therefore, there is a strong desire to make scanned documents digitizable. In the next subsections, we introduce two different software products that perform this task. In addition, techniques for improving the output quality are presented.

3.2 Performance Metric

Natural Language Toolkit (NLTK) Python package ranks among the most widely used libraries dealing with natural language processing topics.
NLTK provides an English dictionary, which is the foundation to calculate the performance of the used algorithms. We compare the number of words that do exist in the mentioned dictionary. An absolute accuracy value is then given by the ratio of the correctly recognized words from the total of the string chains produced by the algorithm.

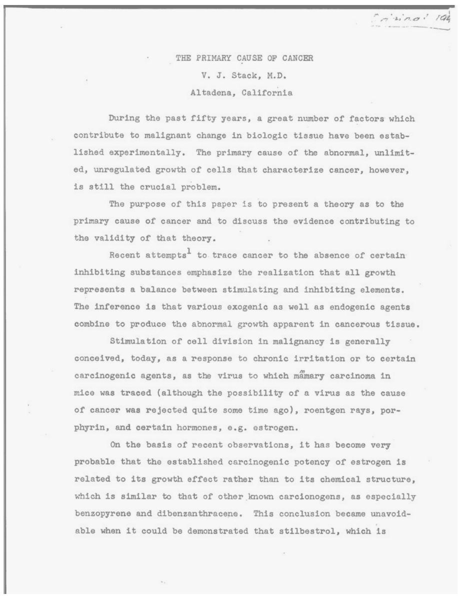
3.3 Preprocessing

One of the issues faced by OCR-software is that the input document images do not necessarily have a good quality. Therefore, we decided to use three different techniques to increase the quality of the images. Our aim here, is to illustrate the power of such tools to the reader. We

are going to discuss *contrast*, *sharpness* and *upscaling* techniques.
Let us start with contrast and sharpness. We used PIL (see Chapter 2), which provides the necessary pre-implemented functions. The description of these functions was taken from [31].

ImageEnhance.Contrast(image): Creates an enhancement object for adjusting contrast in an image. A factor of 0.0 gives a solid grey image, factor 1.0 gives the original image.

We provided a demonstration of the contrast function in Figure 13. Particularly, one can see the original image (13a) and the image after using a contrast filter with factor 2 (13b).



[illegible]

THE PRIMARY CAUSE OF CANCER
V. J. Stack, M.D.
Altadena, California

During the past fifty years, a great number of factors which contribute to malignant change in biologic tissue have been established experimentally. The primary cause of the abnormal, unlimited, unregulated growth of cells that characterize cancer, however, is still the crucial problem.

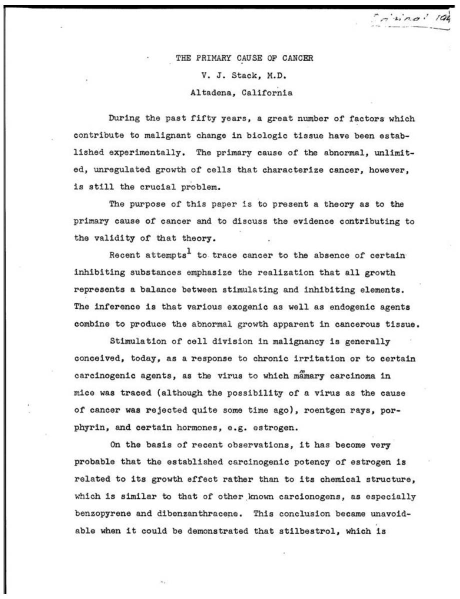
The purpose of this paper is to present a theory as to the primary cause of cancer and to discuss the evidence contributing to the validity of that theory.

Recent attempts[1] to trace cancer to the absence of certain inhibiting substances emphasize the realization that all growth represents a balance between stimulating and inhibiting elements. The inference is that various exogenic as well as endogenic agents combine to produce the abnormal growth apparent in cancerous tissue.

Stimulation of cell division in malignancy is generally conceived, today, as a response to chronic irritation or to certain carcinogenic agents, as the virus to which mamary carcinoma in mice was traced (although the possibility of a virus as the cause of cancer was rejected quite some time ago), roentgen rays, porphyrin, and certain hormones, e.g. estrogen.

On the basis of recent observations, it has become very probable that the established carcinogenic potency of estrogen is related to its growth effect rather than to its chemical structure, which is similar to that of other known carcionogens, as especially benzopyrene and dibenzanthracene. This conclusion became unavoidable when it could be demonstrated that stilbestrol, which is

(a) Pure Image



[illegible]

THE PRIMARY CAUSE OF CANCER
V. J. Stack, M.D.
Altadena, California

During the past fifty years, a great number of factors which contribute to malignant change in biologic tissue have been established experimentally. The primary cause of the abnormal, unlimited, unregulated growth of cells that characterize cancer, however, is still the crucial problem.

The purpose of this paper is to present a theory as to the primary cause of cancer and to discuss the evidence contributing to the validity of that theory.

Recent attempts[1] to trace cancer to the absence of certain inhibiting substances emphasize the realization that all growth represents a balance between stimulating and inhibiting elements. The inference is that various exogenic as well as endogenic agents combine to produce the abnormal growth apparent in cancerous tissue.

Stimulation of cell division in malignancy is generally conceived, today, as a response to chronic irritation or to certain carcinogenic agents, as the virus to which mamary carcinoma in mice was traced (although the possibility of a virus as the cause of cancer was rejected quite some time ago), roentgen rays, porphyrin, and certain hormones, e.g. estrogen.

On the basis of recent observations, it has become very probable that the established carcinogenic potency of estrogen is related to its growth effect rather than to its chemical structure, which is similar to that of other known carcionogens, as especially benzopyrene and dibenzanthracene. This conclusion became unavoidable when it could be demonstrated that stilbestrol, which is

(b) Contrast=2

Figure 13: Demonstration of the contrast function

By changing the contrast factor, the luminance is changed, whereby contours can be better distinguished. This helped our OCR-algorithm perform better.

ImageEnhance.Sharpness(image): Creates an enhancement object for adjusting the sharpness of an image. The factor 0.0 gives a blurred image, 1.0 gives the original image, and a factor of 2.0 gives a sharpened image.

In the next demonstration, we want to show the impact of the sharpness filter on specific areas. To see the changes, we take different areas and zoom in with four dissimilar magnifications into the image. Figure 14 shows the original image.

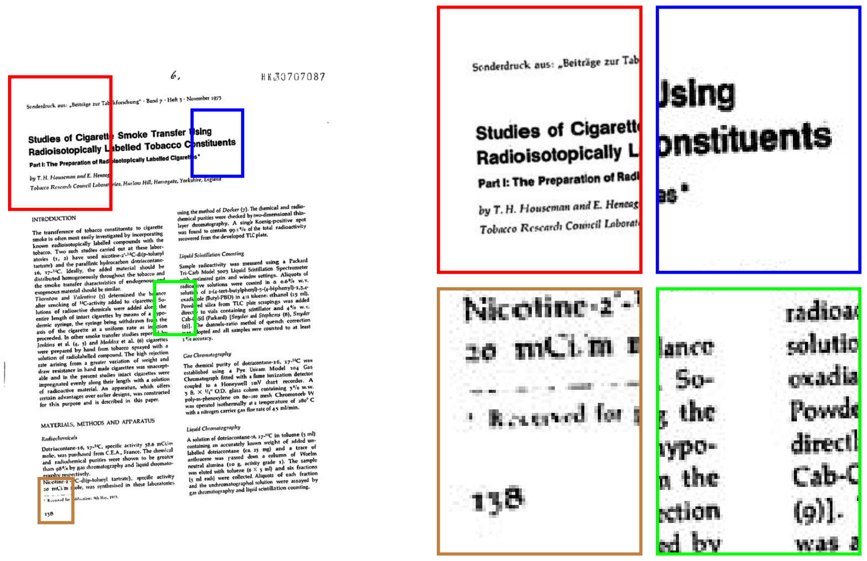


Figure 14: Pure image

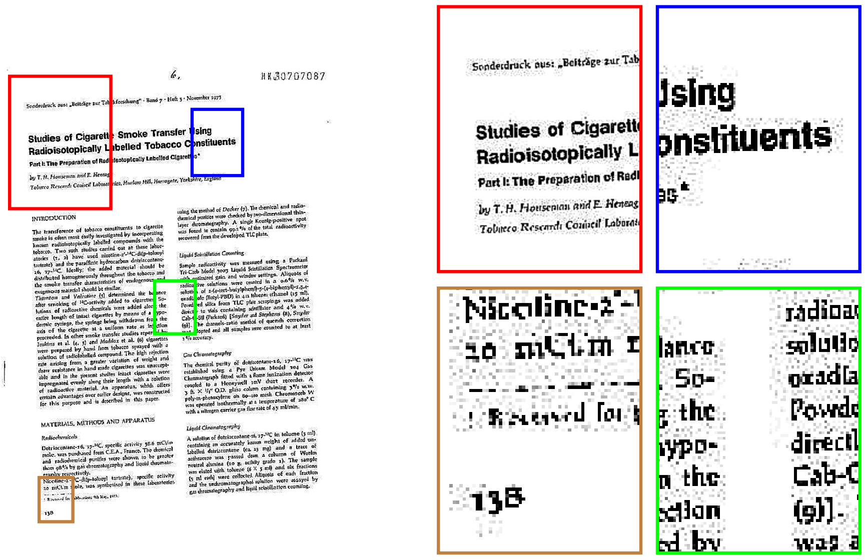


Figure 15: Sharpness=2

Now, let us look into the sharpened one (Figure 15) and compare the areas to the original image. We observe that the pixels are less blurry than in the original image. However, around the words, there is also an increasing of shadows. These shadows could decrease the performance of the OCR-software. We noticed that some images require stronger/weaker contrast and sharpness factors.Therefore, it is not optimal to use fixed sharpness and contrast factor for our images. Each image needs a different parameter configuration to obtain a good quality for the OCR-Software. An enhanced image can be reached by optimizing the contrast and sharpness factors. Hence, we implemented a function `optimal_params` to determine a fairly good configuration. Among a discrete set of factor configurations, we choose those that maximize the accuracy of our algorithm (see Figure 16), defined in 3.2.

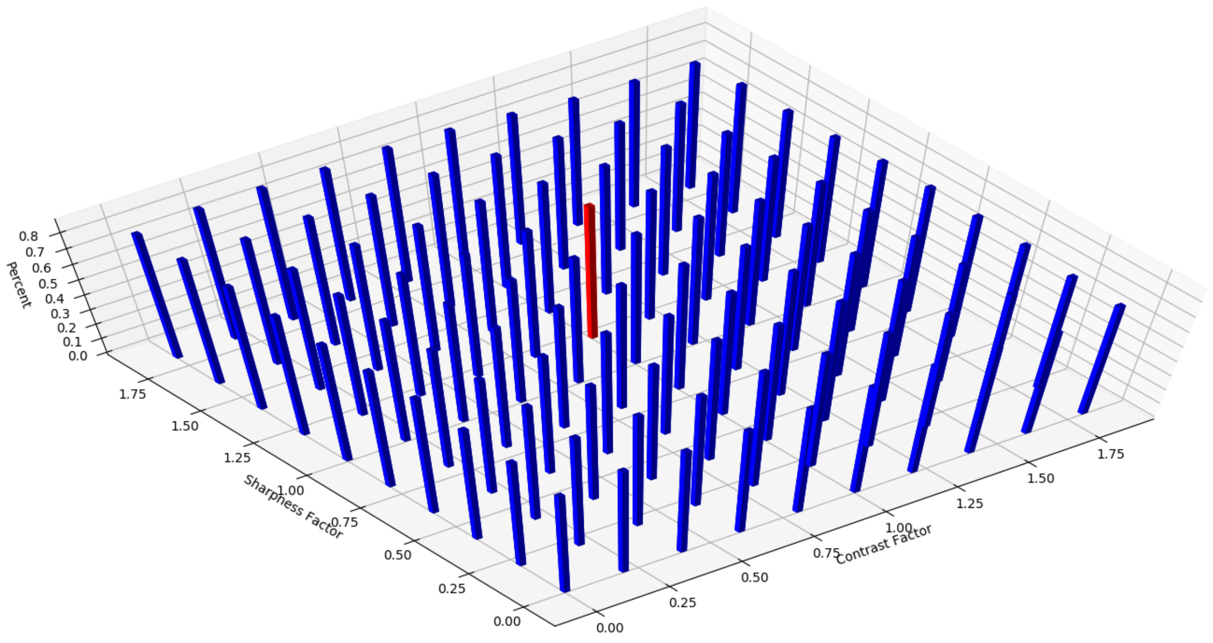


Figure 16: Aerial view: Interaction between contrast and sharpness

The bars in the figure represent the accuracy in percentage of a randomly picked image with different contrast and sharpness factors. The step size in contrast and sharpness direction is of 0.2. In the cloud of blue bars, we have one red bar which shows the maximum accuracy. To see this, take a look at the front view Figure 17.

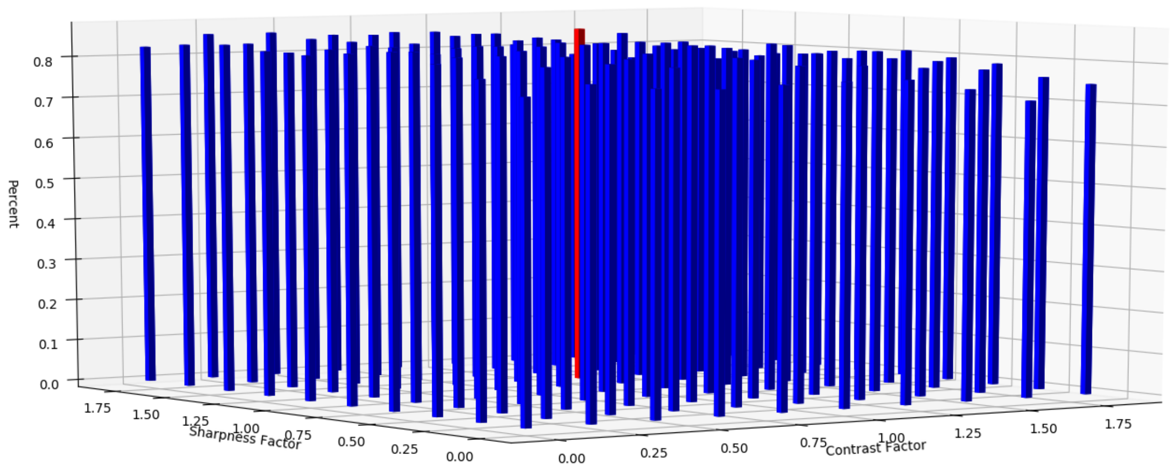


Figure 17: Front view: Interaction between contrast and sharpness

The optimum was reached at a contrast factor equal to 1 and a sharpness factor equal to 1.2. In conclusion, this image is already in a good shape, because the factors are not far away from 1. This image may not require such optimization. However, our dataset contains images for which this procedure is strongly recommended.

Now, we want to discuss the upscaling technique. For our tasks, we have fixed the height and width by twice those of the original image and used a bicubic interpolation. Note that, there are many different methods for interpolation available, for example nearest neighbour, linear, bilinear, cubic interpolation etc. Bicubic interpolation [21] is an extension of cubic interpolation. It interpolates data points in a two-dimensional regular grid. In contrast to other interpolation techniques, bicubic interpolation considers 4×4 pixels and has therefore smoother transitions between pixels and creates less interpolation artifacts. Bicubic interpolation is more computationally expensive than the others, but that was no problem thanks to the LRZ cluster.

3.4 Optical Character Recognition Algorithm

In this section, we briefly explain the algorithm behind the OCR module and the pre-implemented algorithms we tested during the project.

3.4.1 Tesseract-OCR

After inquiring the available algorithms performing OCR and comparing them, we chose to base our OCR micro-service on Tesseract-OCR [46] [51]; an OCR engine supported by Google. Apart from the positive user reviews, Tesseract-OCR has several advantages when it comes to its usability and scalability. It is an open source project developed under the Apache license and can therefore be deployed totally locally and avoid any data privacy concerns for Capgemini. It also supports 6 languages, including German. Tesseract-OCR 4.0, the latest Tesseract-OCR version, also supports multi-columned text, text on images and mathematical equations. Furthermore, Tesseract-OCR can be easily called through PyTesseract, a Python library.

PyTesseract: (The Python Library) Tesseract-OCR has a Python library that makes the call of the OCR engine straightforward. We have to pass an image to the predefined function and we obtain a string of the found characters.

3.4.2 OCR.Space

For the sake of comparison we chose to use OCR.Space [34]. OCR Space is a free cloud-based service that has a callable but limited API. The API is restricted to 500 calls/day and to 25.000 calls/month. The bigger issue with the OCR Space API is the limitation to the file size to 1MB and that data is sent to a remote server which could be problematic to Capgemini because of data privacy issues.
We chose not to include the OCR.Space module in the final OCR micro-service to avoid size and data issues.

3.4.3 OCR.Space VS PyTesseract

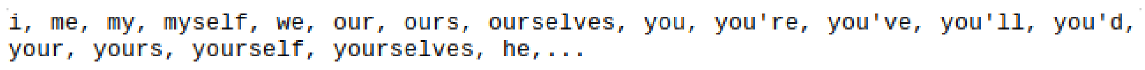
While developing the OCR micro-service, we tested the performance of both algorithms. We noticed that OCR.Space delivered better results than PyTesseract in transforming the input images to plain text. However it took more time to process the files, which is probably mainly caused by the file transfer to and from the cloud.

3.5 Postprocessing

To post-process the outcome of OCR, we used an *auto-corrector* called *TextBlob*, which, among other functionalities, acts as spelling corrector. That works well for general texts, where only a few named entities such as names, locations etc. appear.
However, our dataset includes emails, letters and other classes, where such named entities are likely to occur frequently. For instance, we found out that names are sometimes converted into

vocabulary words, which causes issues with the *Named Entity Recognition* (NER) part, Chapter 4. Therefore, we decided to remove this postprocessing part and leave it optional for the user to activate. The same holds for the two upcoming postprocessing elements we implemented.

Stopwords usually refer to the most common words in a language that hold few to no information in a sentence. Here is a set of examples of stopwords out of the NLTK package:



```
i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd,
your, yours, yourself, yourselves, he,...
```

Sometimes it makes sense to remove such words, e.g. for classifying documents based on text.

Most of *Punctuation characters* transport also no information for certain tasks and could be removed. A possible approach is, to allow only certain characters and that can be done with the Python package *re* (regular expression).

4 Named Entity Recognition (NER)

After performing OCR on the scanned documents and classifying them, we are interested in further information contained in the documents. The goal is to label important words via NER and assign them to certain pre-defined tags. After examining different state-of-the-art methods for NER, we decided to analyze and implement the paper [32] by Ma and Hovy.
Figure 18 places the NER module within our project pipeline. The NER module is intended to be used after the OCR module generating an output for our web application. The output from the OCR module has to be converted into a certain structure, which will be described later. After that, we run the NER model, from which we can either generate an output for our search server or a visual output for the user. Besides, we can feed any text to the NER module and use it as an independent tagging unit.

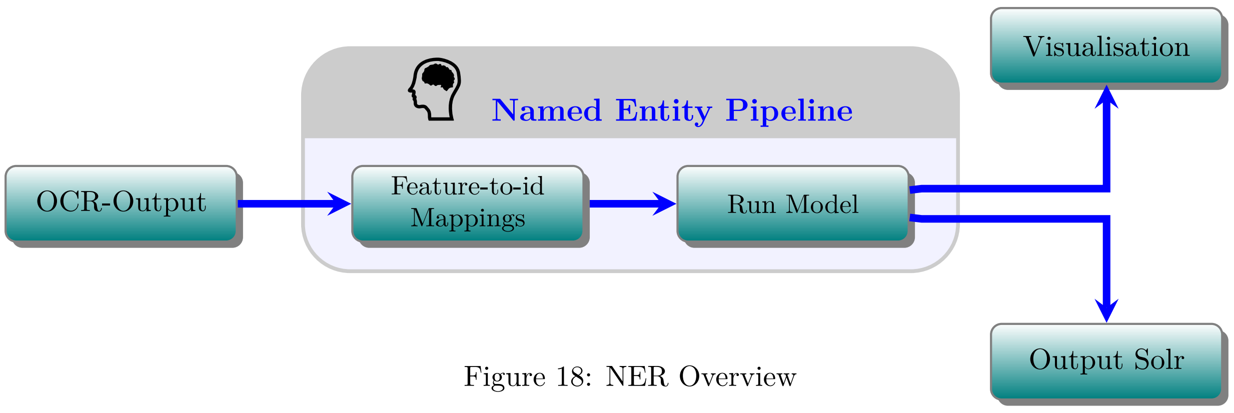


Figure 18: NER Overview

4.1 Problem Definition

The goal is to find out, whether a word belongs to a certain class from a set of predefined classes. Furthermore, the algorithm should be able to correctly classify compound words.

Example (Expectation from NER) *Assume we are interested in the classes {Person, Organization, Location} and want to tag the following sentence with a NER-algorithm:*

'Max Mustermann studies at Technical University of Munich and lives in Munich.'

The expected output of our algorithm is the following:

- *'Max Mustermann' as* one *person*
- *'Technical University of Munich' as* one *organization and*
- *'Munich' as location.*

In this example we already see a potential difficulty. We expect the algorithm to label the 'Munich' in 'Technical University of Munich' as part of the organization and not as location. So the algorithm should also analyze the context of each word.

4.2 Tagging Systems

There are several tagging systems to encode the output of a NER algorithm. It is important to know, whether a word is at the beginning, in the middle or at the end of a sequence of words which belong to one entity. We used the *inside, outside ,beginning* (IOB) tagging system to represent the position of a word in a sequence belonging to one entity. '**B**' means that a word is the beginning of a chunk of words belonging to an entity and '**I**' means that a word is inside a chunk. '**O**' shows that none of the predefined classes correspond to that word. For more details we refer to Appendix A.1 and [38].

Example (Comparison IOB and IOBES) *Let us again consider the same sentence from the previous example and encode the desired NER-output with both tagging schemes as presented in Appendix A.1.*

Table 4: Comparison of tagging schemes

Token	IOB	IOBES	Entity
Max	B	B	Person
Mustermann	I	E	Person
studies	O	O	Other
at	O	O	Other
Technical	B	B	Organization
University	I	I	Organization
of	I	I	Organization
Munich	I	E	Organization
and	O	O	Other
lives	O	O	Other
in	O	O	Other
Munich	B	S	Location

If we are interested whether a word is the end of a chain of words belonging to one entity then

- *for **IOB** we have to look at the next word in the sequence and determine if it is labeled with **B** or **O***
- *for **i**nside **o**utside **b**eginning **e**nd **s**ingletion (**IOBES**) we only have to check if the word is labeled with **S** or **E**.*

Similarly we check whether a word is the beginning of a chain of words belonging to the same entity.

In our case, we worked with the IOB system, since we are bounded to the training set that uses this tagging system.

4.3 Neural Networks vs Classical Statistical Algorithms

In the search of a suitable algorithm for the NER-part of our project we studied the following four papers: [6], [24], [32] and [44]. These papers use similar neural networks to perform NER, yet with slight modifications. We decided to implement the proposed architecture of [32] by Ma and Hovy (with slight modifications according to [39]) as it uses the most sophisticated network and reaches better results compared to the other papers.
We were aware of other algorithms that use non-neural network approaches such as *hidden markov models* [33] or *rule based algorithms* [12]. We decided to use [32] in our work because the proposed network perfectly coincides with our learning goals, as it combines a CNN, a *bidirectional long short-term memory network* (BiLSTM) and a *conditional random field* (CRF) component. These are both mathematically and conceptionally appealing for all of our team members. Moreover, it performs better than the other investigated methods.

4.4 Dataset: CoNLL2003

To train our network we use the free collection of newswire articles from the Reuters Corpus provided by *Conference on Computational Natural Language* (CoNLL) from the shared task from 2003 (CoNLL2003) [40]. The dataset consists of 22137 sentences in total and of 302811 tokens, i.e. words and punctuation characters. Moreover, the dataset is split into a train, validation and test sets which contain 67.6%, 17.0% and 15.4% of the whole dataset respectively. As we can see in the following example, CoNLL2003 is presented as a 4-column plain text file. Only the first (the tokens) and the fourth (Named Entity Tag) columns are relevant for our use-case. The second column represents a part-of-speech tag (e.g. noun, verb etc.) while the third column contains syntactic chunk information. For instance, consider the example in Table 5.

Table 5: CoNLL2003 Example

Token	POS-Tag	SCI	NER-Tag
U.N	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-PP	I-LOC
.	.	O	O

4.5 Model Architecture: Embeddings-CNN-BiLSTM-CRF

The model we selected concatenates pre-trained *word embeddings*, trainable *character representations* and trainable *casing information*. First, to achieve a character representation, we fed a CNN with input dimension 30×50 and window size 3, where 30 is the dimension of the character embedding and 50 is the number of characters per word after padding. The CNN then outputs a representation of dimension 30 for each word. The word embedding of dimension 100 together with the output of the CNN and a 10 dimensional casing information vector constitute the input to a BiLSTM which outputs a 200 dimensional vector for each word. The output of the BiLSTM layer then is input to the CRF, which computes the most probable sequence of entities for one sentence. In the following subsections we explain the parts of the model architecture in more details.

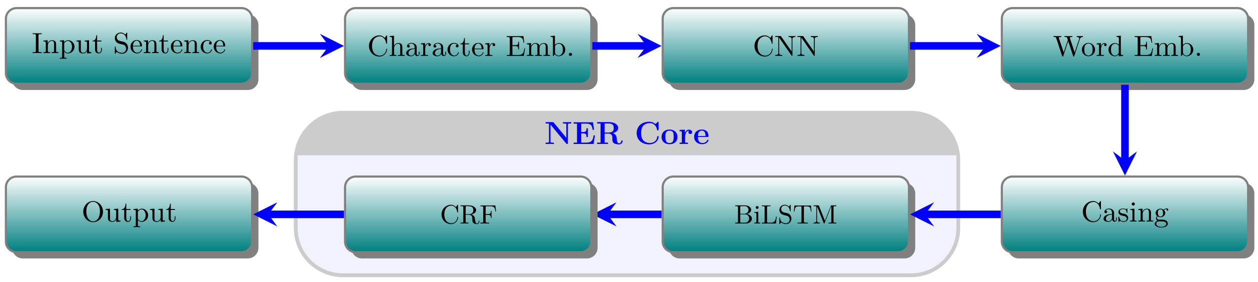


Figure 19: NER - From sentence to output

4.5.1 Word Embeddings

We use the *GloVe* pretrained word embeddings provided by Stanford [37]. This dataset consists of 6 billion tokens from Wikipedia and from newswire [2] data which were embedded in $\mathbb{R}^d$, $d \in D := \{50, 100, 200, 300\}$. We trained our algorithm for all dimensions $d \in D$. According to [32], the GloVe embeddings yield better results compared to other known word embeddings under the proposed model [32].

4.5.2 Character Representation and Casing Information

As mentioned in Subsection 4.5, we want to use character representations for the words to extract more fine-grained information than with the word embeddings. Therefore, we first pad every word to a fixed length of 50 characters and then embed the characters into a d-dimensional array ($d = 30$ in [32]) using a trainable embeddings layer. These embedded characters serve as input to the above mentioned CNN, which then, coupled with a *Max Pooling layer*, gives us a character representation of dimension 30 for every word as shown in Figure 20.

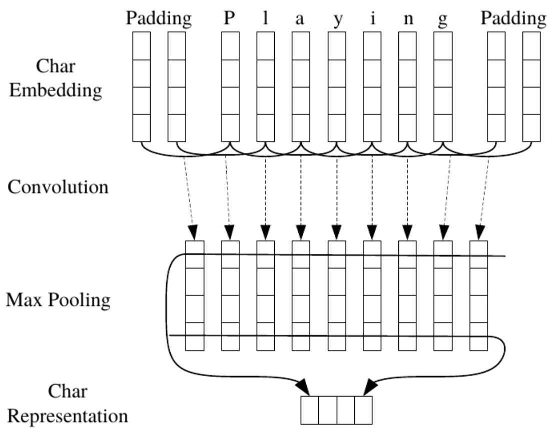


Figure 20: From words to character representation [32]

Furthermore, similar to [32], we use casing information of the words as input to the BiLSTM part of our network. We consider whether a word:

1. is a digit
2. is mainly numeric. That is the case when the ratio of characters in the words which are digits and the length of the word is larger than 0.5 and smaller than 1
3. is completely lower case and has no digit information from the above two cases
4. is completely upper case and the above conditions do not apply
5. only the first letter is capital and above conditions fail
6. none of the above.

As mentioned above, the casing information is embedded into $\mathbb{R}^{10}$ as trainable parameters of the model.

4.5.3 Bidirectional Long Short-Term Memory

Until now, we described how we embed our available characters and casing information in higher dimensions of $\mathbb{R}$ and represent the words as an output from a CNN. To use these embeddings we concatenate the word embeddings, character representations and casing information into one vector. After this step, we will be able to feed a whole sentence into the BiLSTM, a special type of *recurrent neural networks* that prevents the *vanishing and exploding gradient problems* [5], [17].

BiLSTMs are useful when the data can be modeled sequentially or is timely distributed. As the contexts of the sentences are of particular importance when searching for entities, a BiLSTM architecture is suited due to its ability to consider left and right information of a word. Consider the above example where it is important to consider the left side of 'Munich' from 'Technical University of Munich' to be able to differentiate 'Munich' from the entity type location.

A BiLSTM is composed of a *backward LSTM* that looks at the left side of a word and a *forward LSTM* that considers the right side of a word. Moreover, a (Bi)-LSTM, in contrast to RNNs, controls how much information is let through to the next state and how much information gets into the actual state [17]. Using the same notation as in [32], we can define an LSTM-network as follows:

Definition (LSTM cell [32]) *Let x_t be the concatenated embeddings at input cell t as explained above, i.e the t-th word belonging to one sentence. Let σ be the sigmoid function and h_t be the hidden state at input state t. A LSTM cell consists of several gates which control the (amount of) information flow. In particular, there is the forget gate which controls the amount of information getting in one cell from the previous state, the input gate which controls the amount of update for the cell and the output gate which controls the amount of output. Defining U_i, U_f, U_c, U_o to be the weight matrices for the input at time t and $W_i, b_i, W_f, b_f, W_c, b_c, W_o, b_o$ the weight matrices and bias vectors for the hidden state at time t we obtain for the update formulae for state t:*

$$\begin{aligned}
i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) \\
f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\
\tilde{c}_t &= \tanh(W_c h_{t-1} + U_c x_t + b_c) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \\
o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \\
h_t &= o_t \otimes \tanh(c_t) \qquad (4.1)
\end{aligned}$$

where $\otimes$ defines the entry-wise multiplication.

Figure 21 shows the above computations of one LSTM cell state t.

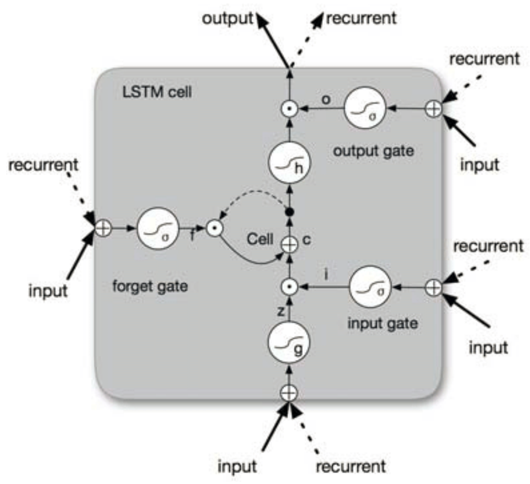


Figure 21: LSTM cell at state t [32]

4.5.4 Conditional Random Field

The CRF is the final building block of the whole architecture, as shown in Figure 22. It outputs a sequence of entities and operates on one single sentence at a time. The CRF allows us to jointly consider the correlations of the labels of a word between the words on the right and the ones on the left. The CRF computes the most probable sequence of labels for a sentence. The task of the CRF is to maximize the log-likelihood of the *conditional probability* [32]

$$p(y|z;W,b) = \frac{\prod_{i=1}^{n} \exp(W^T_{y_{i-1},y_i} z_i + b_{y_{i-1},y_i})}{\sum_{y' \in \Omega(z)} \prod_{i=1}^{n} \exp(W^T_{y'_{i-1},y'_i} z_i + b_{y'_{i-1},y'_i})}$$

w.r.t to the trainable parameters W and b where

- $z = (z_1, \dots, z_n)$ is a vector of the concatenated feature embeddings (word, character, casing) of a sentence
- $y = (y_1, \dots, y_n)$ the sequence of labels for z
- $\Omega(z)$ is the set of all possible label chains for the sentence z
- $W_{x,y}$ represents the weight vector for two labels x and y
- $b_{x,y}$ is the bias corresponding to the two labels x and y.

Having the optimal weight vector W and bias b we can choose the label sequence v by

$$v \in \operatorname*{argmax}_{y \in \Omega(z)} p(y|z;W;b) \tag{4.2}$$

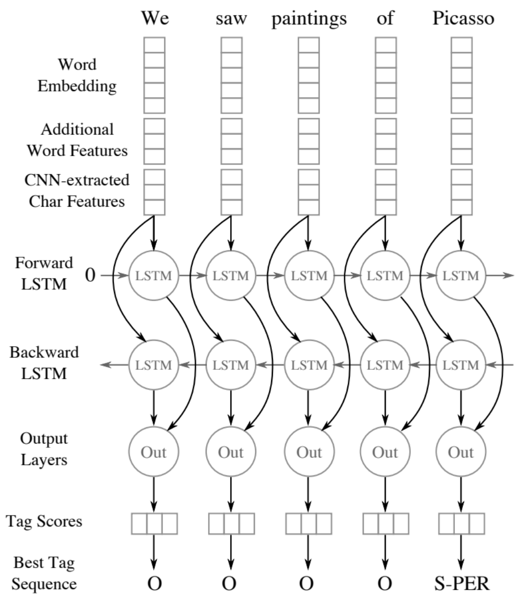


Figure 22: Complete architecture [7] with IOBES tagging

4.6 Implementation

We mainly used the algorithm implemented by the paper [39] and we therefore, forked the corresponding repository [23] and continued to work on that. We deployed the model in Python 3.6 together with Tensorflow and Keras. Keras allowed the easier use of embedding layers, CNNs, LSTMs and BiLSTMs. That means that we have to assign a unique number to every word embedding from GloVe and we need to create a mappings dictionary which stores distinctly the 'word-to-id' information, the 'character-to-id' information, the 'casing-to-id' and the 'named-entity-to-id' information. Furthermore, we worked with a data dictionary which contains a train matrix, a test matrix and a validation matrix. These three matrices are generated from the train, test and validation files from the CoNLL2003 dataset. The three matrices are similar: For instance, the train matrix contains a dictionary for each sentence which contain the ids for one sentence, each word split into its representation with character ids, the casing ids, the IOB encoding of the labels from CoNLL2003 dataset again transformed into ids and the raw sentence.

Example (Key Mappings) *Let us have a look at one (half) sentence stored in the train matrix of the data dictionary*

```
{'CoNLL2003':
    {'trainMatrix':
        [{'tokens': [192629, 11, 35],
          'casing': [6, 4, 4],
          'NER_IOB': [0, 0, 0],
          'characters' : [[41, 27, 26, 18, 21, 16, 17, 26, 15, 17],
                           [21, 26],
                           [32, 20, 17]]
          'raw_tokens': ['Confidence', 'in', 'the']}]}}
```

This dictionary contains now the (half) sentence 'Confidence in the', whereas 'Confidence' has id 192629, 'in' has id 11 and 'the' has id 35. The casing information are that 'Confidence' has only *an upper case in the first character and 'in' and 'the' have no casing information. Furthermore, in the 'characters' key is each word split into its characters with their respective ids.*

After that we can plug these id information into the model starting with a non trainable embeddings layer for the word embeddings and a trainable embeddings layer for the character embeddings and casing embeddings followed by a CNN layer for the character representation. After that we can apply the LSTM (forward and backward) layers and finally the CRF classifier.

4.7 Training

We used dropout and *recurrent dropout* [42] of rate 0.25 for both, forward and backward, LSTMs. We trained on randomly shuffled batches with approximately 32 sentences in each batch. Furthermore, we used the Adam optimizer for the model with default parameters [22] together with clip norm 1 to control *gradient clipping* to prevent gradient explosions [19]. The character embeddings are initialized with uniform samples in $\left(-\sqrt{\frac{3}{d}}, +\sqrt{\frac{3}{d}}\right)^d$, where d is the character embedding dimension [32], [39]. Training was performed on the LRZ Linux Cluster as explained in Subsection 2.7.

4.8 Model Performance

We need a metric to measure how well our classifier labels each word. Moreover, the CoNLL2003 dataset is heavily unbalanced [40], i.e. the training set contains approximately 88% tokens that are labeled with 'O'. So imagine if we defined an extremely biased classifier toward this label. The classifier would label all tokens as 'O' and the final accuracy on the training set would be very high. However, it may be that the classifier has not captured any entities outside of 'O'. For this reason we decided to use the so called *F1-measure* since it is able to capture the accuracy across all labels. For the definition of the F1-measure (harmonic sum of recall and precision) and the definition of the recall and precision we refer to Appendix A.2. With this underlying definition of a measure for accuracy, we now compare results for our model.

4.8.1 Training with Different Word Embedding Dimensions of GloVe

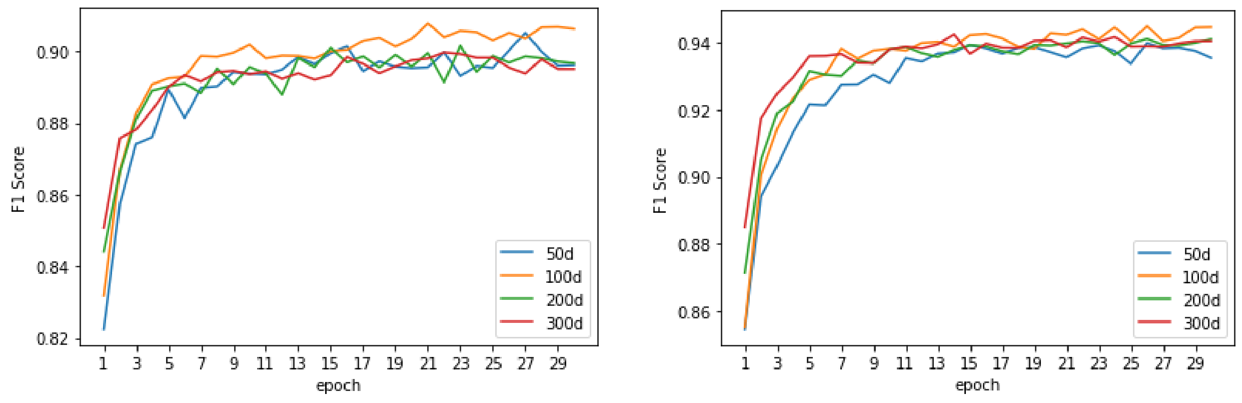


Figure 23: F1 scores for different word embedding dimensions for test (left) and validation (right)

We trained with different dimensions of the GloVe word embeddings to examine the effects of the embedding dimension on the model performance. As we can see in the F1-scores in Figure 23, the 100-dimensional GloVe embeddings perform better, both for the test and the validation sets.

The model achieved its maximal score of 90.8% for the test set on epoch 21 with a validation score of 94.2%.

4.8.2 Training with No Casing Information

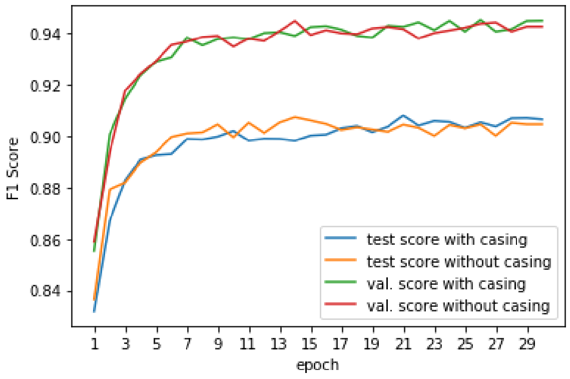


Figure 24: Comparison of F1-scores for test and validation

We tested whether the casing information we take into account contributes to the performance of the model when we use 100 dimensional word embeddings. We see in Figure 24 that the F1 scores for the test and validation dataset are barely distinguishable. However, we notice a difference of around 4% between the validation and test accuracy. This is comparable to the results found in the original paper. The difference in validation and test was expected, since we have fine-tuned our model hyper-parametrs and architecture (word embeddings dimensions) based on the validation data. A difference of 4% is not uncommon in this context.

4.8.3 Training with Different Character Embeddings Dimensions

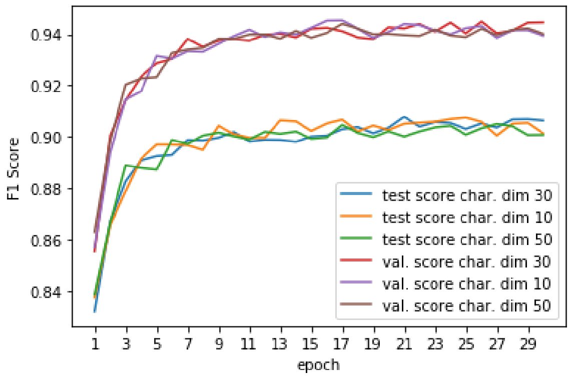


Figure 25: Comparison of F1 scores with different character embedding dimensions for test and validation

In our model, we also needed to define an embedding dimension for the characters to be able to feed the CNN. Therefore, we also experimented with the dimension of these embeddings with fixed 100 dimensional word embeddings and 10 dimensional casing information embeddings. We see that there are barely no differences in the model performance and again, we notice a difference of 4% between the validation and test accuracy. The same interpretation as above applies here as well.

Recall and Precision metrics that have been defined together with the F1-measure in 4.5, are plotted as well in the same section in the Appendix.

5 Entity Search Tool & User Interface

The goal is now to provide the end-user with a *web interface* for their scanned documents. They should be able to search within found named entities with the web interface. To achieve this we used *Solr* and *Banana*.

5.1 Solr

Solr is a free and open-source software under the terms of Apache license [14]. Furthermore, it is a Java search server which makes it possible to search for the named entities within the scanned documents. These documents need to be first transformed into a certain JSON format.
The search server searches a so-called *index*. The procedure of searching an index can be compared to the procedure of searching a list of keywords at the end of a book. Once the keyword is found, we open the corresponding page in the book. The same happens in the server, it serves the user with the JSON file containing the right index [48].

To allow the server to perform the index searches, we must first index the JSON files. The indexing procedure may be compared to the procedure of creating an index for keywords in a book. The content extracted by indexing the JSON files is stored in a so called *Core*. It is an instance of a class called *Lucene index* [53]. An instantiated core contains configuration files such as `managed-schema.xml`, that are later modified by the user. For an instantiated core, one has to add the structure of the JSON files before indexing it.
In our case, we added to the file `managed-schema.xml` the needed *fields* that describe the structure of the JSON-files. Basically, a field is an abstraction of the key-value pair defined in JSON.

5.2 Set Up Solr as Service

Figure 26 depicts our steps to set up Solr as a service for the end-user.

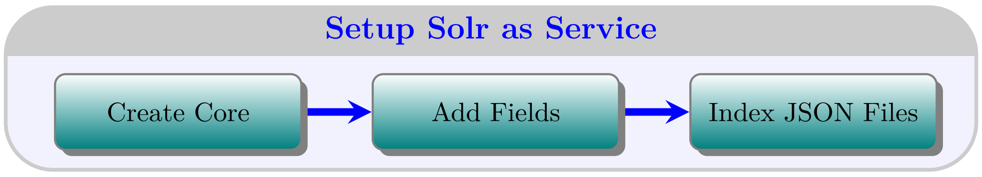


Figure 26: Solr setup

5.3 Set Up Banana as Service

Banana sits on top of Solr and provides a web interface that can connect to a specific core. The web interface is presented as a dashboard. Configuring the dashboard for the needs of an end-user can be conveniently done with a few mouse clicks.
Banana dashboard consists of panels [30]. We decided to use the *query panel*, *table panel* and *hits panel*. In the query panel, the search entities are typed, e.g. the user might search for a person that was mentioned somewhere in the emails. The hits panel gives the total number of found JSON files, i.e. the number of emails containing the looked-up person, and the table panel shows the found files in a readable manner.

5.4 JSON Structure and Banana Dashboard

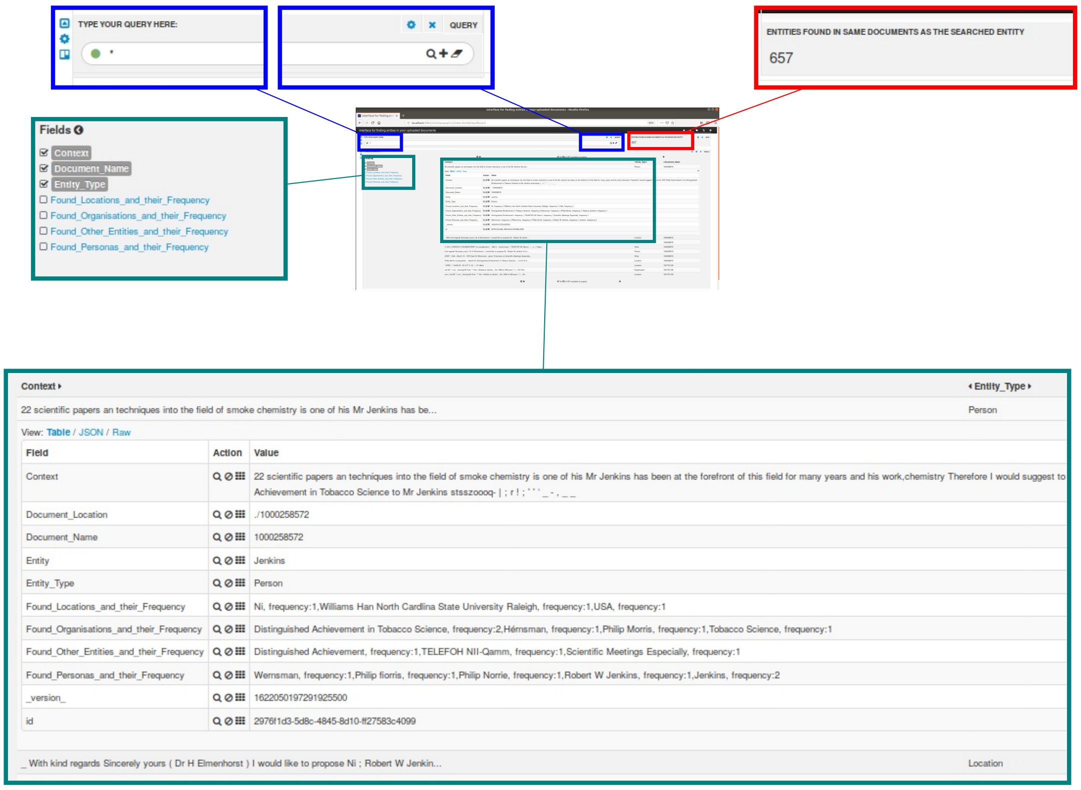
The challenging part has turned out to be the appropriate design of the JSON structure that is fed into the Solr search server. We tried to find a JSON structure that meets the expectations of the product owner, Capgemini: We needed to visualize the required information in the dashboard. However, after trying many different JSON structures and tweaking the source code of the Banana dashboard, we were not able to find a solution that satisfied all the requirements. The reason was that Solr depends heavily on the JSON structure that could not capture every information for the required visualizations.
The final JSON structure is shown in Listing 1. The open issue here consists in the hits panel that does not show the correct number of found documents which contain the searched entity.

```
[{"Entity": name of found entity,
"Context": the context where the entity appeared,
"Document_Name": name of the document,
"Document_Class": class of the processed .txt document,
"Found_Personas": list of personas and their frequencies
   extracted from the document,
"Found_Locations": list of locations and their frequencies
   extracted from the document,
"Found_Organisations": list of organisations and their
   frequencies extracted from the document,
"Found_Other_Entities": list of important entities and their
   frequencies extracted from the document
}, ... , {...}]
```

Listing 1: Re-worked JSON Structure

5.4.1 Final Banana Dashboard

In Figure 27 we can see our final web interface provided to the end-user. The blue frame represents the query panel. For more information on how to formulate queries, please refer to Appendix B. The red frame is supposed to display the number of scanned documents that contain the searched entity. However, it now shows the total number of entities that were found within the documents that contain the searched entity. The green frames represent the table panel: they display the found JSON files in a readable way.


TYPE YOUR QUERY HERE:
QUERY
ENTITIES FOUND IN SAME DOCUMENTS AS THE SEARCHED ENTITY
657
Fields
Context
Document_Name
Entity_Type
Found_Locations_and_their_Frequency
Found_Organisations_and_their_Frequency
Found_Other_Entities_and_their_Frequency
Found_Personas_and_their_Frequency

Context	Entity_Type
22 scientific papers an techniques into the field of smoke chemistry is one of his Mr Jenkins has be...	Person

View: Table / JSON / Raw

Field	Action	Value
Context	🔍 ⊘ ▦	22 scientific papers an techniques into the field of smoke chemistry is one of his Mr Jenkins has been at the forefront of this field for many years and his work,chemistry Therefore I would suggest to Achievement in Tobacco Science to Mr Jenkins stsszoooq- | ; r ! ; ' ' ' _ - , _ _
Document_Location	🔍 ⊘ ▦	./1000258572
Document_Name	🔍 ⊘ ▦	1000258572
Entity	🔍 ⊘ ▦	Jenkins
Entity_Type	🔍 ⊘ ▦	Person
Found_Locations_and_their_Frequency	🔍 ⊘ ▦	Ni, frequency:1,Williams Han North Cardlina State University Raleigh, frequency:1,USA, frequency:1
Found_Organisations_and_their_Frequency	🔍 ⊘ ▦	Distinguished Achievement in Tobacco Science, frequency:2,Hérnsman, frequency:1,Philip Morris, frequency:1,Tobacco Science, frequency:1
Found_Other_Entities_and_their_Frequency	🔍 ⊘ ▦	Distinguished Achievement, frequency:1,TELEFOH NII-Qamm, frequency:1,Scientific Meetings Especially, frequency:1
Found_Personas_and_their_Frequency	🔍 ⊘ ▦	Wernsman, frequency:1,Philip fiorris, frequency:1,Philip Norrie, frequency:1,Robert W Jenkins, frequency:1,Jenkins, frequency:2
version	🔍 ⊘ ▦	1622050197291925500
id	🔍 ⊘ ▦	2976f1d3-5d8c-4845-8d10-ff27583c4099

_ With kind regards Sincerely yours (Dr H Elmenhorst) I would like to propose Ni ; Robert W Jenkin...	Location

Figure 27: Final Banana Dashboard

6 Web Application

Once all the pipeline modules are successfully built and tested, the next step is to connect the different services and guarantee the usability of our product.
The connection of the different project components was built as a *RESTful* web application using *Flask*, together with Python, where every component runs on an independent *virtual machine* (VM). The VMs are created and monitored by the *OpenNebula* cloud infrastructure provided by LRZ.

6.1 Computing Infrastructure

OpenNebula is a cloud computing platform on LRZ, which allows customers to upload and create their own VM images. These images can contain different *operating systems* or serve as a *storage device*. For our virtual machines, we use *Ubuntu 16.04 LTS*, which is a Linux-distribution based on Debian.

6.1.1 Infrastructure on OpenNebula

To make our system powerful and consistent, we decided to distinguish between two kinds of VMs. On the one hand, there is the `Filesystem` which acts as memory location for the raw and processed data. This VM is connected to all other VMs in the system [Figure 28].

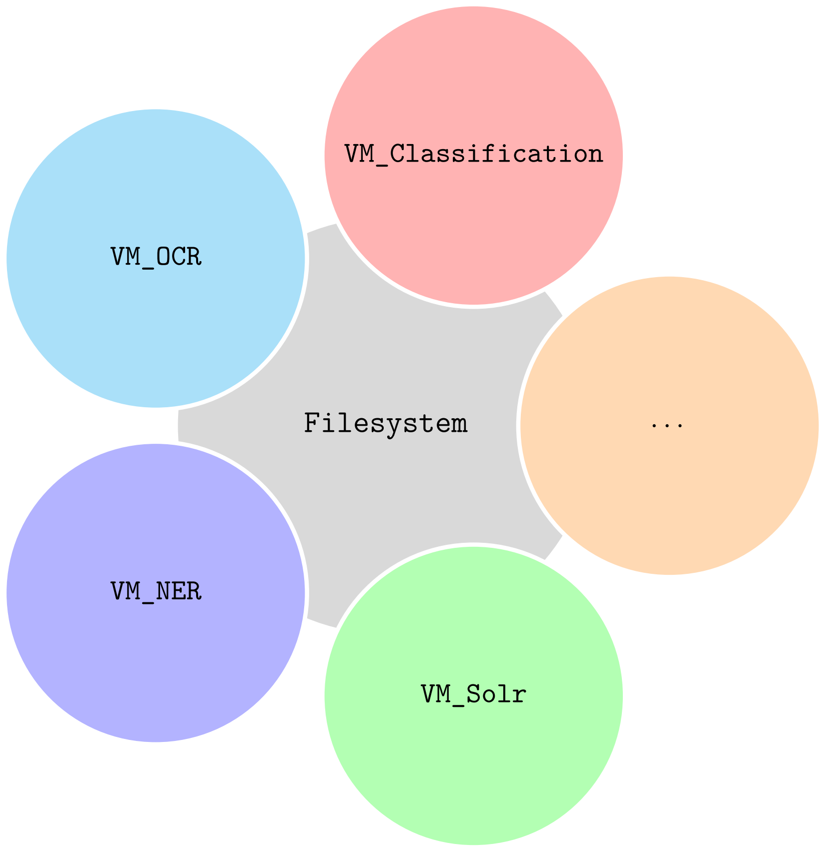


Figure 28: Interaction of VMs

On the other hand, we have the *clients*. In our system there are four different clients that are mounted to `Filesystem`. The mount is required to read and write data from and to the

`Filesystem`. The client then executes a specific job with this data and stores the processed data to a new directory on the server. Clients in our associated system are named after their particular jobs: `VM_Classification`, `VM_OCR`, `VM_NER` and `VM_Solr`.
We provided more information about the `Filesystem`, Client and an instruction sheet for OpenNebula in Appendix C.1, C.2 and C.3 respectively.

6.1.2 Troubleshooting

We would like to point out some of the issues we have had with the OpenNebula cloud service.

When we started using OpenNebula, we had trouble with the amount of available resources. Our cloud had already been used by other groups working with Capgemini. Thus we had fewer resources than we needed.

We also had some problems with the set up of our cloud infrastructure. The provided documentation to the OpenNebula cloud by LRZ is outdated and therefore, confusing and has some missing components. Therefore, we had to get in touch with the LRZ support quite often, which brings us to the next point.

The LRZ support was always friendly and tried to help us with questions. But the long waiting times for an answer and the limited support at the weekends have disturbed our workflow.

Lastly and most relevant, the VMs are fragile in the BOOT and SHUTDOWN processes. This was frustrating because we had to contact the LRZ support each time and wait for an intervention to be able to get access back to our VMs.
The computing nodes accessible with OpenNebula operate with old CPU's which was problematic since we used Tensorflow v1.12.0 until we wanted to deploy our services on OpenNebula VMs. Unfortunately, the old CPU's in the cloud could only support up to Tensorflow v1.5.0, which meant a lot of extra work and debugging.

6.2 Web Application Microframework

When designing the final product of this project together with Capgemini, we decided to modularize the whole pipeline. Each module runs on a separate virtual machine as explained above to guarantee the independence of the micro-services.
This requires a steady communication between the different virtual machines and that is where Flask and the RESTful API come in handy.

6.2.1 The RESTful API

The different micro-services of our project communicate through a RESTful-API: an API that follows the representational state transfer (REST) architecture. The API uses the *Hypertext Transfer Protocol* (HTTP) [13] governing the GET, PUT, POST and DELETE requests.

6.2.2 Flask & Flask-restful

Flask is a Python-based *web microframework*. It provides the means to build and deploy a web application, i.e. a computer program stored and running over a remote server. Together with flask-restful, a Flask extension, it covers all the tools required to guarantee an instant communication protocol between our different micro-services using the predetermined requests to manage data: the GET, POST and DELETE requests.

6.3 The Web Application Building Blocks

In this part, we explain the communication protocol between the different VMs.

6.3.1 Master node server

The master is the code entity that controls the flow and the processing of data. It uses the POST request to start a certain micro-service once the data is ready. It is connected to all the components. It can be started from anywhere.

6.3.2 Micro-service servers

Each micro-service is assigned to a separate Flask server defined as a Python class and running on the same VM. Each server is started and keeps listening to the master node requests. Once a POST request is sent, the corresponding data file is transmitted to the right VM using a GET request sent to the `Filesystem`.
Once the job is done, the processed data, i.e. the output of the micro-service, is sent to the `Filesystem` using a POST request. The communication is, however, exclusively done with the master node.

7 Conclusion

Many decisions within companies rely on information extracted out of documents. In many cases the key information has to be extracted by hand. To simplify the information extraction we built a pipeline that automates key information extraction from (scanned) documents.
In this project we delivered an implemented pipeline that matches the solution architecture designed together with Capgemini. The pipeline consists of different micro-services: Image Document Classification, Optical Character Recognition and Named Entity Recognition. We built an ensemble network that, based on region-specific and holistic representations of a document, classifies the document type.
As expected, the ensemble network has reached satisfactory results on two types of classes. Furthermore, with Tesseract, we were able to apply OCR on our scanned documents. By changing the contrast, sharpness and size of the scanned documents, we have improved the number of correctly recognized characters.
Moreover, with BiLSTM, CNN and CRF we successfully deployed the NER module. With NER we label relevant entities within documents such as locations, organizations and people.
To make the found entities searchable and the output of the NER module visually convenient, we used Solr and Banana. Finally, with Flask, we run the pipeline as a web application that was deployed using OpenNebula.

Outlook

A possible fine-tuning of the pipeline could be to make use of the postprocessing implementations *after* the NER tagging and thereby avoid the auto-correction of named entities non-existing in the English dictionary. This would improve the output of the OCR module.
We can further increase the performance of the OCR module by performing a more sophisticated search procedure for the optimal parameters than the one presented in Subsection 3.3, e.g. random search or using bayesian optimization.
The classification micro-service can be easily retrained to classify more than two classes. Therefore, the pipeline is able to accept more different types of documents. By increasing the number of classes we expect that the test accuracy might decrease. As a counter-measure, we propose the document clustering technique [3] or the *intra-domain transfer learning* used in [9]. Intra-domain transfer learning consists in performing transfer learning from the trained holistic CNN to the region-based CNNs and counts as an 'inter-domain' operation since the training data are basically the same.
Y. Shen et al. [44] have proposed an amelioration to the NER module proposed by [32]. They suggest to use *active learning* [43] as a means of reducing training time and especially overcoming the issue of new training data that requires labeling. The main idea behind it, is to actively choose the examples to first annotate from the chosen dataset, use it to fine-tune the model and let the model choose the next round of data to be annotated (based on how uncertain the model is about their labels). This training loop is then repeated until a desired model performance is attained.
Finally, due to lack of time, we were not able to put *Jenkins*, our *Continuous Integration* (CI) tool to work. Nevertheless, we have prepared it and all that still needs to be done, is to write to test scripts and integrate it with our Gitlab repository. This would allow future teams that work on our project to automate the testing procedure and generate test reports live, while working on the project and therefore, avoid bigger coding issues and conflicts.

Appendices

A Named Entity Recognition

A.1 Tagging Systems

Definition (Tagging Systems [38]) *Let $n \in \mathbb{N}$ and $C := \{L_1, \dots, L_n\}$ be a set of classes we want to use to tag a sequence of words $S := (W_1, \dots, W_m)$, $m \in \mathbb{N}$, with.*

- *The **IOB** (inside outside beginning) tagging scheme is used to encode, whether a word is at the beginning or inside an entity or does not belong to any $L \in C$, i.e there are $i_1, \dots, i_m \in \{\text{'I', 'O', 'B'}\}$ and $j_1, \dots, j_m \in \{1, \dots, n\}$ such that W_k is in class L_{j_k} and is the beginning of a chain of words belonging to one entity if $i_k =$ 'B', inside a chain if $i_k =$ 'I' or does not belong to any class if $i_k =$ 'O'.*

- *The **IOBES** (inside outside beginning end singleton) tagging scheme is an extension of the **IOB** scheme as it further characterizes words. With this extension we can also encode when words are singletons, only one word belongs to an entity, and if a word is the end of a chain of words belonging to one entity.*

A.2 Model Performance

Definition (F1-Measure) *If class of a word, not 'O', has been classified correctly we call it a true positive (tp), If a class of a word, not 'O', has been classified wrong we call it a false negative (fn). If a word with class 'O' has been classified wrong we call it a false positive (fp). The precision P, the recall R and the $F1$-measure F of a classifier are defined as*

$$P := \frac{tp}{tp + fp}, \quad R := \frac{tp}{tp + fn}, \quad F := 2 \cdot \frac{P \cdot R}{P + R}.$$

A.2.1 Precision and Recall for Different Word Embedding Dimensions of GloVe

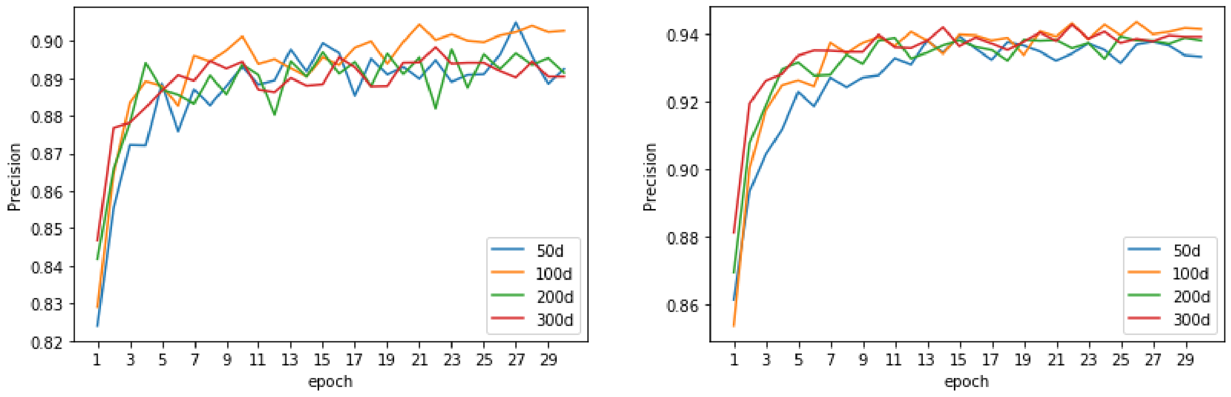


Figure 29: Precision for different word embedding dimensions for test (left) and validation (right)

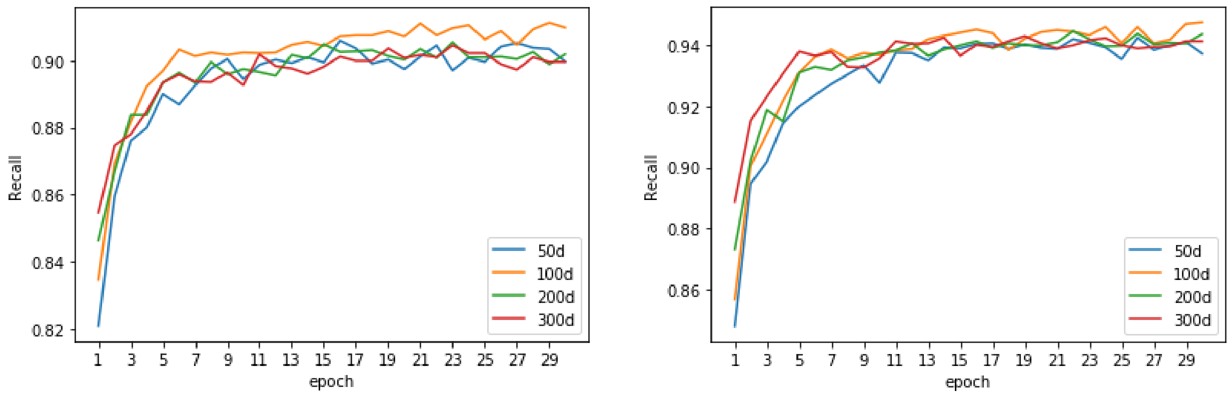


Figure 30: Recall for different word embedding dimensions for test (left) and validation (right)

The precision and recall of the test and validation dataset in the Figures 29 and 30 show similar behaviour compared to the scores with the F1 measure in Figure 23 with both datasets.

A.2.2 Precision and Recall for Training With No Casing Information

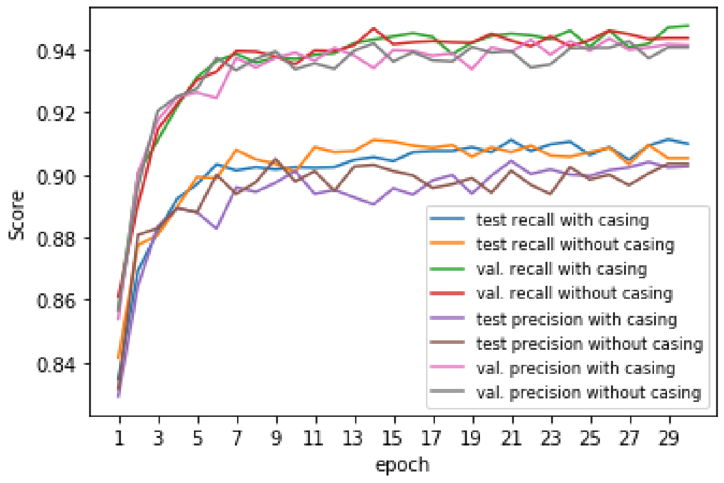


Figure 31: Comparison of precision and recall with and without casing information

A.2.3 Precision and Recall with different character embeddings dimensions

For the precision and recall, Figure 32, one can not make clear distinctions between the three casing dimensions due to similar behaviour of the scores.

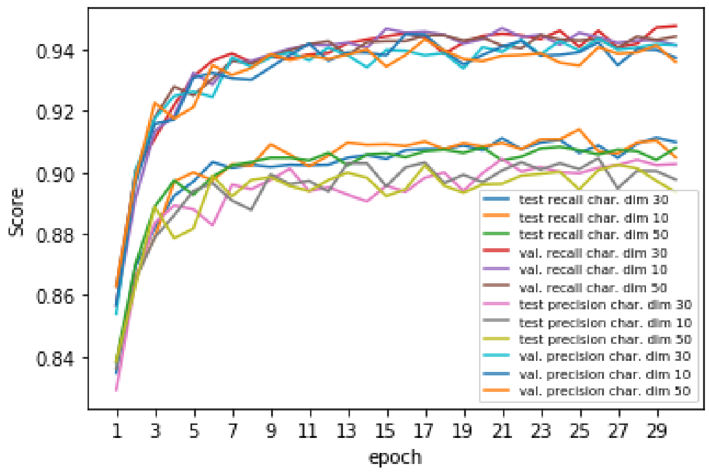


Figure 32: Comparison of precision and recall for different character embedding dimensions

B Formulating Solr Queries

With Solr we are able to search within indexed JSON files. In the following we demonstrate how to perform searches.
Assume we are looking for a person named Winston within our JSON files. To formulate a query that returns all files containing an entity named Winston, we type in the query panel:

```
"Winston"
```

This might return files that contain Winston as an organization or a place. To obtain files that contain Winston as a person we execute:

```
(Entity:"Winston") AND (Entity_Type:"Person")
```

The `AND` operation means that the found files must contain both terms.
To find files that contain Winston, but not the entities Germany or France:

```
((Entity:"Winston") AND (Entity_Type:"Person")) -("Germany" OR "France")
```

The minus sign in front of `("Germany" OR "France")` means logical not. Further, the `OR` operation means that one of the terms must be contained in the files. Thus, Solr will return files not containing the term France or Germany.
There are many other possible query operations that Solr offers [49]. However, the above presented operations are sufficient to cover basic needs to search within files.

C OpenNebula

C.1 Server (`Filesystem`)

Let us take a deeper look into the `Filesystem`. Additionally to the Ubuntu image, a datablock image runs on the `Filesystem`, which provides us with an extra amount of storage. For our needs, we decided to add a 500GB datablock. To use such an image, we had to mount it in to the VM. This can be done as follows:

1. Start the VM in the OpenNebula application.
2. Open a Terminal in Ubuntu and connect to your VM:

   ```
   ssh root@my-server-IP
   ```

3. Create a shared folder (storage) and mount it to the datablock image (vdd), execute:

   ```
   mkdir storage
   mkfs -t ext3 /dev/vdd
   mount /dev/vdd /root/storage
   ```

4. To have a persistent mount, execute

   ```
   vim /etc/fstab
   ```

   and add into fstab the following line:

   ```
   /dev/vdd  /root/storage    ext3    defaults     0       0
   ```

Now, the datablock image is persistently mounted to the VM. In the next step, we install Network File System (NFS) for the server. NFS is a network protocol, which allows you to exchange data into a local network. Execute:

```
sudo apt-get update
sudo apt-get install nfs-kernel-server
```

We also want that VMs starting with IP's 10.155.208.xxx or 10.155.209.xxx are able to mount to the server. For that reason, execute

```
vim /etc/exports
```

and add into exports the following two lines:

```
/root/storage   10.155.208.0/24(rw,sync,no_subtree_check,no_root_squash)
/root/storage   10.155.209.0/24(rw,sync,no_subtree_check,no_root_squash)
```

After a change in exports execute always:

```
sudo exportfs -a
```

Before we are able to mount a client to the server, we need changes in the firewall settings. You can check the status of the firewall by executing

```
sudo ufw status
```

The predefined status should be inactive. To activate the firewall execute

```
sudo ufw enable
```

and add the required changes:

```
sudo ufw allow from 10.155.208.0/24 to any port nfs
sudo ufw allow from 10.155.209.0/24 to any port nfs
sudo ufw allow ssh
```

To make our changes persistent run:

```
sudo systemctl disable netfilter-persistent
```

C.2 Client

The setup of a client is straightforward compared to the server. Note that, the server has a fixed IP address in contrast to the clients. The subsequent procedure must be done for each client separately. So, start the VM in the OpenNebula application. Afterwards, start a Terminal in Ubuntu and connect to your VM with:

```
ssh root@my-client-IP
```

At first, we install NFS for the client. This is usually done by:

```
sudo apt-get update
sudo apt-get install nfs-common
```

Let us create a folder called filesystem:

```
mkdir filesystem
```

Assume that 10.155.208.226 is the IP of the related server. To have a persistent mount to the server, we need to add into fstab the following line:

```
10.155.208.226:/root/storag /root/filesystem nfs
    auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

Mounting the client to the server can be done by:

```
sudo mount 10.155.208.226:/root/storage /root/filesystem
```

C.3 Instruction Sheet for OpenNebula

To work with the previous created setup without problems, please follow the belated steps:

1. Log in to the portal: https://www.cloud.mwn.de/login
   -use your LRZ identification

2. In the left-hand menu, click on Instances and then on VMs.

3. Mark the checkbox of `Filesystem` next to the VM and press the start button. After a couple of seconds refresh the website. The status of your VM should be now on RUNNING. The IP address of the `Filesystem` is fixed.

4. Unmark the checkbox of `Filesystem` and mark one of the clients, e.g. `VM_Classification`. Press the start button and after a couple of seconds refresh the website. The status should be also on RUNNING. Keep in mind the IP address of the started VM.

5. Open a Terminal on your Ubuntu machine. Execute:

   ```
   ssh root@<IP-address>
   ```

   Enter your Password.
   Now you have full access to the VM.

The following receipt gives instructions on how to shut down a VM.

1. Open the Terminal, where you run your VM.

2. Execute:

   ```
   logout
   ```

3. Go back to the OpenNebula application

4. In the left-hand menu, click on Instances and then on VMs.

5. Mark your RUNNING VM and click on the shut down button on the top to click: Undeploy

6. Now your VM has shut down and is not using resources. This procedure has to be done always and is necessary to save resources.

The `Filesystem` should be started always before the clients. Only that way we can guarantee that the mount between the VMs is set correctly.

D Web App In Use

In this section, we will exhibit the functional logic behind our Flask app. For the sake of simplicity, we restrict the explanation to the communication between the flask master node and the OCR host server.
The first part includes configuring the name and the address of the OCR host server. This will later on allow the master node to request and post data to the intended service.
Calling

```
python ocr\_server.py
```

will start the OCR server on the predetermined address. The server is based on the OCR_Controller class:
1. Define the OCR_Controller: A high level API that calls the OCR_Worker object. It requires the fileclient object that is called in its obj.process(id) method. The id refers to the image to be processed. The latter is streamed through fileclient.get_file method. OCR_Controller stores the output using the fileclient.postfile method.
2. Define the OCR_worker: Python object with the needed attributes and attribute functions to perform OCR. Most importantly it has the method obj.process(image) that returns the output string.
Once the OCR_controller is up and running, we can post requests for the OCR worker to process using

```
requests.post(<url of the to-be-processed image>)
```

The main assumption for the pipeline to work is that all images are saved under ./img/ folder and have an unified data type (e.g. .png, .tif etc.).

References

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Robert Parker et al. “English Gigaword Fifth Edition LDC2011T07”. In: (2011).

[3] Nicholas Andrews and Edward A. Fox. “Recent Developments in Document Clustering”. In: 2007.

[4] Andrew D. Bagdanov and Marcel Worring. “Fine-grained document genre classification using first order random graphs”. In: *CoRR* (2001). URL: https://ieeexplore.ieee.org/document/953759.

[5] Y. Bengio, P. Simard, and P. Frasconi. “Learning Long-term Dependencies with Gradient Descent is Difficult”. In: *Trans. Neur. Netw.* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181. URL: http://dx.doi.org/10.1109/72.279181.

[6] Jason P. C. Chiu and Eric Nichols. “Named Entity Recognition with Bidirectional LSTM-CNNs”. In: *CoRR* abs/1511.08308 (2015). arXiv: 1511.08308. URL: http://arxiv.org/abs/1511.08308.

[7] Jason P. C. Chiu and Eric Nichols. “Named Entity Recognition with Bidirectional LSTM-CNNs”. In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 357–370.

[8] François Chollet. *keras.* 2015. URL: https://github.com/fchollet/keras.

[9] Arindam Das, Saikat Roy, and Ujjwal Bhattacharya. “Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep Convolutional Neural Networks”. In: *CoRR* abs/1801.09321 (2018). arXiv: 1801.09321. URL: http://arxiv.org/abs/1801.09321.

[10] Hoan Manh Dau and Ning Xu. “Text Document Classification Using Support Vector Machine with Feature Selection Using Singular Value Decomposition”. In: *Advanced Materials Research* 905 (2014). URL: https://www.scientific.net/AMR.905.528.

[11] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09.* 2009.

[12] Tome Eftimov, Barbara Koroušić Seljak, and Peter Korošec. “A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations”. In: *PLOS ONE* 12.6 (June 2017), pp. 1–32. DOI: 10.1371/journal.pone.0179488. URL: https://doi.org/10.1371/journal.pone.0179488.

[13] R. Fielding et al. In: (June 1999). URL: https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

[14] Apache Software Foundation. *A Quick Overview.* 2018. URL: https://lucene.apache.org/solr/guide/7_5/a-quick-overview.html (visited on 02/06/2019).

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. URL: http://www.deeplearningbook.org.

[16] Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. “Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval”. In: *CoRR* abs/1502.07058 (2015). arXiv: 1502.07058. URL: http://arxiv.org/abs/1502.07058.

[17] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[18] Daniel Jurafsky and James H. Martin. “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition”. In: 3rd edition. Stanford University and University of Colorado at Boulder, 2018. Chap. Chapter 6, Logistic Regression.

[19] Sekitoshi Kanai, Yasuhiro Fujiwara, and Sotetsu Iwamura. “Preventing Gradient Explosions in Gated Recurrent Units”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 435–444. URL: http://papers.nips.cc/paper/6647-preventing-gradient-explosions-in-gated-recurrent-units.pdf.

[20] Keras. *Default parameters for Adam*. 2018. URL: https://keras.io/optimizers/ (visited on 02/01/2019).

[21] Robert Keys. “Cubic convolution interpolation for digital image processing. IEEE Trans Acoust Speech Signal Process”. In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 29 (Jan. 1982), pp. 1153 –1160. DOI: 10.1109/TASSP.1981.1163711.

[22] Diedrik P. Kingma and Jimmy Lei Ba. “ADAM: A Method for Stochastic Optimization”. In: *ICLR* (2015). URL: https://arxiv.org/abs/1412.6980.

[23] Ubiquitous Knowledge Processing Lab. *emnlp2017-bilstm-cnn-crf*. 2018. URL: https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf.

[24] Guillaume Lample et al. “Neural Architectures for Named Entity Recognition”. In: *CoRR* abs/1603.01360 (2016). arXiv: 1603.01360. URL: http://arxiv.org/abs/1603.01360.

[25] Leibniz-Rechenzentrum. *Job Processing on the LRZ Clusters*. 2018. URL: https://www.lrz.de/services/compute/linux-cluster/job_processing/ (visited on 02/01/2019).

[26] Leibniz-Rechenzentrum. *Overview of the Clusters Configuration*. 2018. URL: https://www.lrz.de/services/compute/linux-cluster/overview/ (visited on 02/01/2019).

[27] *Logo Flask*. URL: https://de.wikipedia.org/wiki/Datei:Flask_logo.svg (visited on 02/08/2019).

[28] *Logo OpenNebula*. URL: https://opennebula.org/referencing/ (visited on 02/08/2019).

[29] *Logo Solr*. URL: http://lucene.apache.org/solr/logos-and-assets.html (visited on 02/08/2019).

[30] Lucidworks. *Banana*. 2016. URL: https://doc.lucidworks.com/lucidworks-hdpsearch/2.5/Guide-Banana.html#_panels (visited on 02/07/2019).

[31] Fredrik Lundh and Matthew Ellis. “Python Imaging Library Overview”. In: (2002). URL: http://www.pythonware.com/media/data/pil-handbook.pdf (visited on 01/12/2018).

[32] Xuezhe Ma and Eduard H. Hovy. “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. In: *CoRR* abs/1603.01354 (2016). arXiv: 1603.01354. URL: http://arxiv.org/abs/1603.01354.

[33] Sudha Morwal, Nusrat Jahan, and Deepti Chopra. “Named Entity Recognition using Hidden Markov Model (HMM)”. In: *International Journal on Natural Language Computing* 1 (Dec. 2012), pp. 15–23. DOI: 10.5121/ijnlc.2012.1402.

[34] OCR.Space. URL: https://ocr.space/.

[35] Travis Oliphant. *NumPy: A guide to NumPy*. USA: Trelgol Publishing. 2006. URL: http://www.numpy.org/.

[36] Sinno Jialin Pan and Qiang Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22 (2010), pp. 1345–1359.

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

[38] Lev Ratinov and Dan Roth. "Design Challenges and Misconceptions in Named Entity Recognition". In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. CoNLL '09. Association for Computational Linguistics, 2009, pp. 147–155. ISBN: 978-1-932432-29-9. URL: http://dl.acm.org/citation.cfm?id=1596374.1596399.

[39] Nils Reimers and Iryna Gurevych. "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Copenhagen, Denmark, Sept. 2017, pp. 338–348. URL: http://aclweb.org/anthology/D17-1035.

[40] Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Association for Computational Linguistics, 2003, pp. 142–147. DOI: 10.3115/1119176.1119195. URL: https://doi.org/10.3115/1119176.1119195.

[41] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. 2016. URL: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100.

[42] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. "Recurrent Dropout without Memory Loss". In: *CoRR* abs/1603.05118 (2016). arXiv: 1603.05118. URL: http://arxiv.org/abs/1603.05118.

[43] Burr Settles. *Active Learning*. Morgan & Claypool Publishers, 2012. ISBN: 1608457257, 9781608457250.

[44] Yanyao Shen et al. "Deep Active Learning for Named Entity Recognition". In: *CoRR* abs/1707.05928 (2017). arXiv: 1707.05928. URL: http://arxiv.org/abs/1707.05928.

[45] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv* (2014). URL: https://arxiv.org/abs/1409.1556.

[46] R. Smith. "An Overview of the Tesseract OCR Engine". In: (2007). DOI: 10.1109/ICDAR.2007.4376991. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4376991&isnumber=4376969.

[47] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[48] Kelvin Tan. *Basic Solr Concepts*. 2019. URL: http://www.solrtutorial.com/basic-solr-concepts.html (visited on 02/06/2019).

[49] Kelvin Tan. *Lucene Query Syntax*. 2019. URL: http://www.solrtutorial.com/solr-query-syntax.html (visited on 02/09/2019).

[50] Lisa Torrey and Jude Shavlik. "Transfer Learning". In: *Handbook of Research on Machine Learning Applications*. Information Science Reference, 2009. Chap. 11. ISBN: 978-1605667669. URL: http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf.

[51] GitHub US. URL: https://github.com/tesseract-ocr/tesseract.

[52] Alexandra Vettori. "Akte um Akte". In: *SZ* (2019). URL: https://www.sueddeutsche.de/muenchen/freising/neue-gemeindearchivarin-akte-um-akte-1.4300730.

[53] Solr Wiki. *Solr Terminology*. 2013. URL: https://wiki.apache.org/solr/SolrTerminology (visited on 02/06/2019).

[54] David H. Wolpert. “Stacked generalization”. In: *Neural Networks* 5.2 (1992), pp. 241 –259. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80023-1. URL: http://www.sciencedirect.com/science/article/pii/S0893608005800231.