Big Data Content Analytics

Spring Term 2018-2019

Professor: Harris Papagewrgiou

Big Data Content Analytics Final Assignment

AUEB Invoicer

MSc in Business Analytics

September 2019

Kyrgiou Aggeliki

(ID: P2821818)

Konstantinidis Antonios

(ID: P2821816)

Papadoulis Dimitrios

(ID: P2821820)

Table of Contents

List of Figures

Abstract

In the context of the course Big Data in Content Analytics, we undertook a project based on a real business case. We chose to focus on the processing and automation of the invoicing procedure which still remains a major issue for many companies and banks, because each day they have to process a lot of invoices and manually enter the data to their system. In this report, we describe our approach starting from a very simple assumption, that of that most of the times the transactions between two parties in terms of invoicing are based on certain templates of invoicing. Then, we gradually build our application based on that assumption.

Description

As already mentioned in the Abstract section, our team chose to explore the invoicing process. Actually, what we wanted to accomplish was to find a way to somehow automate the process of invoicing, which is currently an open issue in the business world. Many companies and banks are currently trying to find ways of automating the process, but there is not any apparent solution. This is why it is utterly useful and important to explore this topic, because in that way you decrease the probability of mistake by humans and you have better monitoring of your logistics and accounting.

Businesswise, it is very useful as it is one of the core operations that are being executed almost every day, taking into account that the transactions and payments is the base of the modern industrial world. Our application can be a useful service on top of these operations.

Structure

There were three main roles for the development of this project who briefly had the following responsibilities.
Data Scientist who is responsible for:

- Convolutional Neural Network for image classification development
- Feed forward Neural Networks for language prediction development
- Implementation of flask
- Microframework

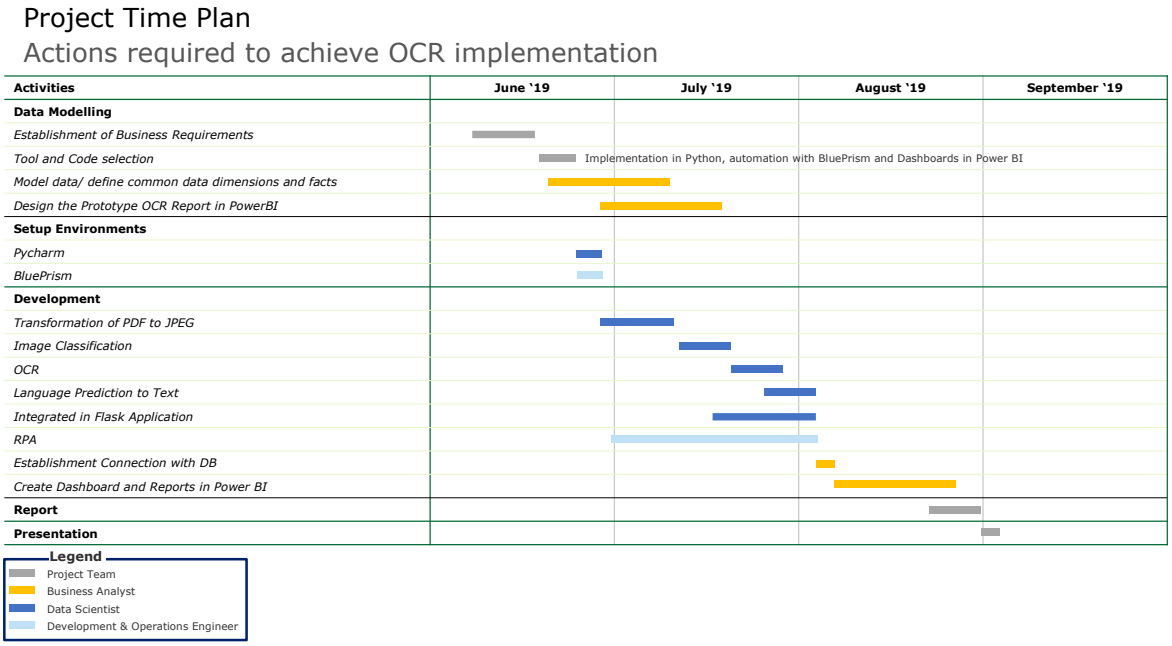
Development and operations engineer who is responsible for:

- Robotic Process Automation
  - Download attached invoices from Outlook
  - Upload Invoices to Flask Application

Business Analyst who is responsible for:

- Database Creation
- Dashboard and Monitoring Reports creation

Project Schedule Planning



5

Mission

Many people have adopted the idea that Optical Character Recognition is a new technology, whereas, in reality the first tries of implementing OCR can be traced back to 1800. Around 1870, an image transmission system using a mosaic of photocells was patented, from which the retina scanner originated. Furthermore, David Shepard is considered as one of the pioneers in OCR because of his crucial role and action by founding the Intelligent Machine Research Corporation in Washington in order to research, develop and build a practical OCR system.
During the following years, more and more firms were involved in OCR technologies, resulting in a large-scale commercialization of OCR systems and equipment during the 70s. Consequently, in 90s OCR was used widely by enterprises in order to automate the management of physical documents such as invoices, receipts, passports, bank statements and any other form of documentation that need to be scanned and transformed in digital file. Even programs like Adobe Acrobat that transforms text to PDF are a way to use OCR and many people have not even realized it.
Nowadays, even if OCR systems are still used to convert physical file (handwritten or printed text) into a digital one, there was still the need for the computer to capture and understand the content of these files. In order to achieve the complete automation of file management, enterprises are trying to combine OCR tools with AI tools. The combination of these tools and the relevant actions implementing by them are depending on the AI platform and the process that needs to be automated.
In conclusion, the combining AI and OCR together gains more and more ground as their impact in terms of the limit of manual activities increases the productivity and reduces the cost and the time.

As already mentioned, we chose to approach the invoicing process. Today one typical invoicing process can be seen below:

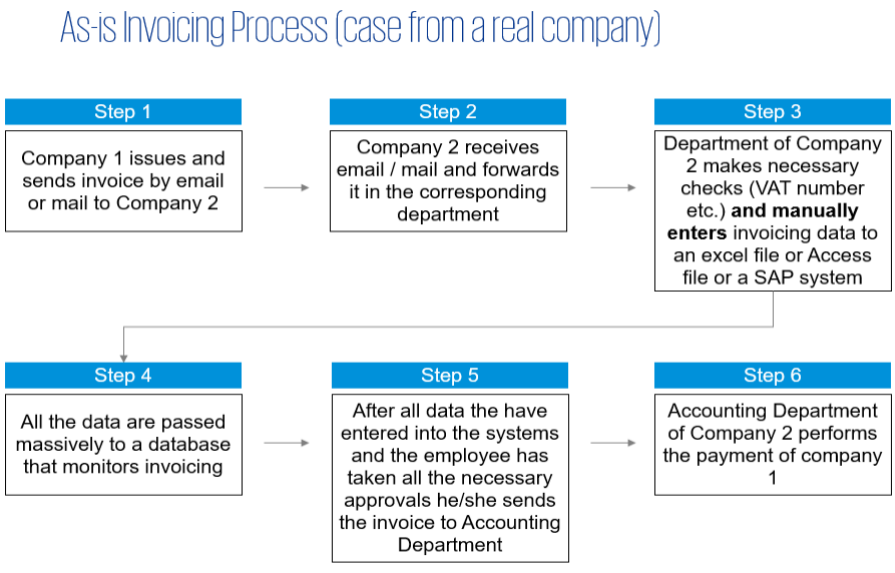


Figure 1: As-is invoicing process

As we can see from Figure 1, an invoice must change a lot of hands and a lot of validation to eventually get paid. But the most important thing is that an employee must enter the data manually to the system. This is where most errors may occur, and this is due to human error. Due to the fact that the companies each day process a very large number of invoices and their respective physical documentation a lot of mistakes can happen. Additionally, an invoice may go through a lot of departments in order to get checked, which may cause delays in the actual payment.

In order to solve this problem, we thought of automating this process. Of course, we have made some assumptions along the road so that our solution can be correct and at the same time adaptable to a real-world scenario. Our application was based on the hypothesis that the invoices that are issued follow a specific format and are based on a certain template by each company. Also, for the current project we kept really simple and we hypothesized that the only values that are needed are the name of the company, the VAT number the total sum and the deadline of the invoice for a payment to be made. The only validating element that we tried to introduce was that of the VAT number, which is unique for an entity or an individual and can be cross validated. We have succeeded on validating the VAT number by connecting to http://ec.europa.eu/taxation_customs/vies/ , which is a site that can tell you if a VAT number for a specific country is valid or not. As far the process itself compared to Figure 1, can be seen below:

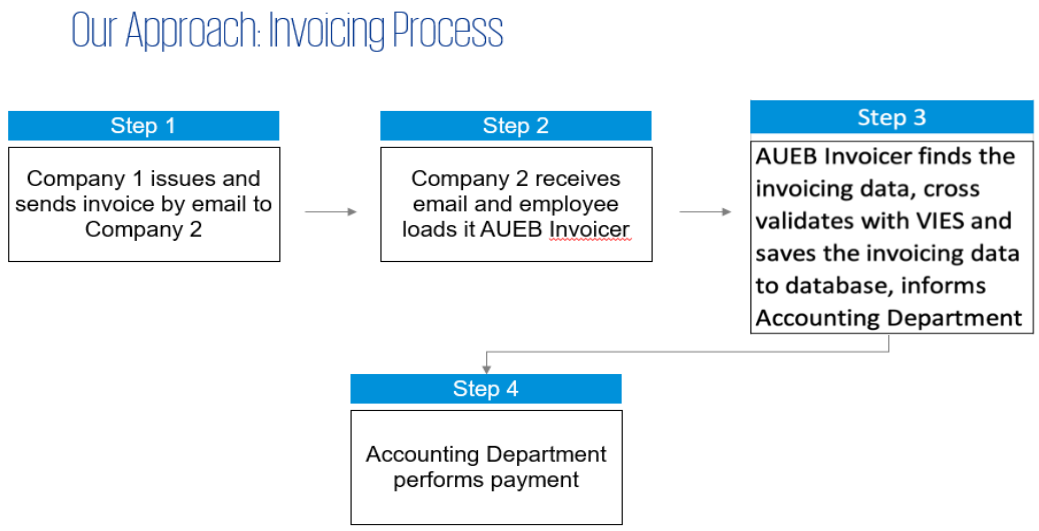


Figure 2: Our approach - AUEB Invoicer

*A sample of the invoice alert can be found at the appendix

Of course, except of decreasing the steps and gaining time, we have made a BI dashboard that lets the employee in charge to inspect the data and see how much the company spends, who are their vendors, to who they owe money etc. An example of this dashboard can be seen below.

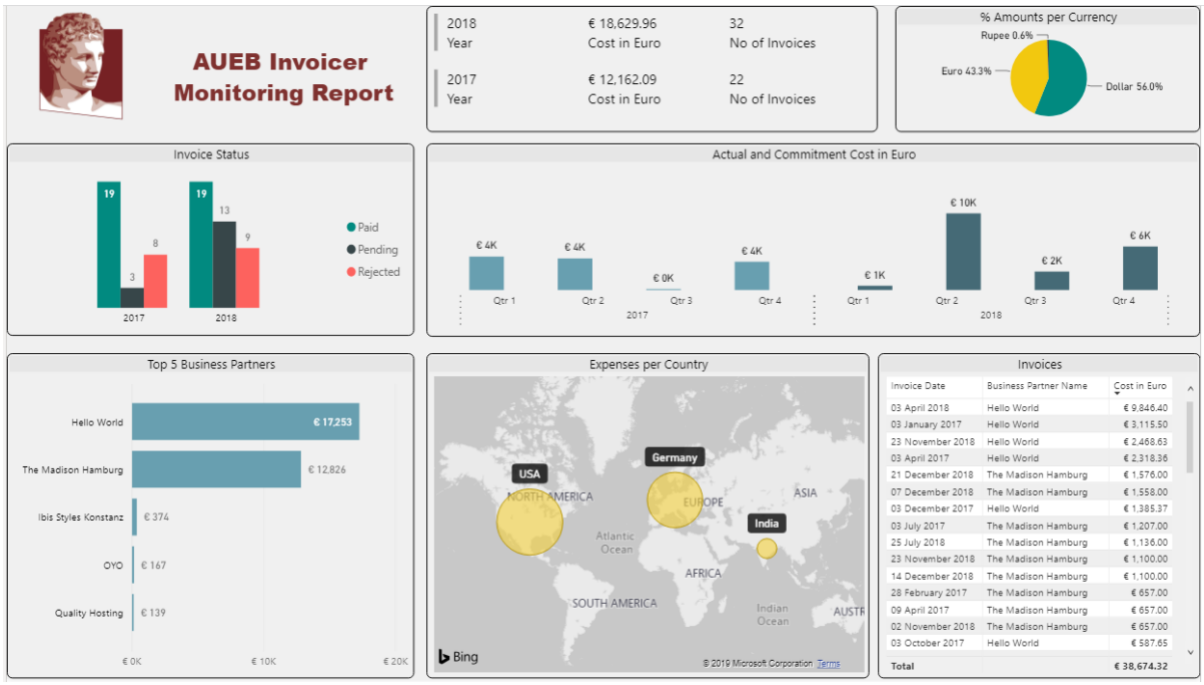


Figure 3: Monitoring Dashboard

Let's know see how we constructed our application before going into detail and talk about the models used.

For our application which was written in Python we used a lot of tools, which are:

1. Flask which is web application framework and in that way we constructed our page.
2. Selenium (for web scrapping and cross validation)
3. Google's API tesseract for OCR in images
4. CV2 for image processing
5. SMTPLIB for sending the email notification
6. Tensforflow and Keras for developing our neural network models, one for language prediction and one for image classification.
7. PYPDF2 and magick wand for processing pdfs and converting them to jpeg format.
8. BluePrism for automation (separate application)

Our application's home page, which can be initiated by going via terminal to the respective folder where the application is located and then run ***python app.py***. After that an http address will appear and then if you copy and paste in any browser, you can see something like this:

Figure 4: AUEB Invoicer Home page

Then, by choosing a testing invoice with the same template of that were used in training, we can easily locate the elements that were mentioned before. The result can be seen below:

Figure 5: Results

Let's now see how our application works:

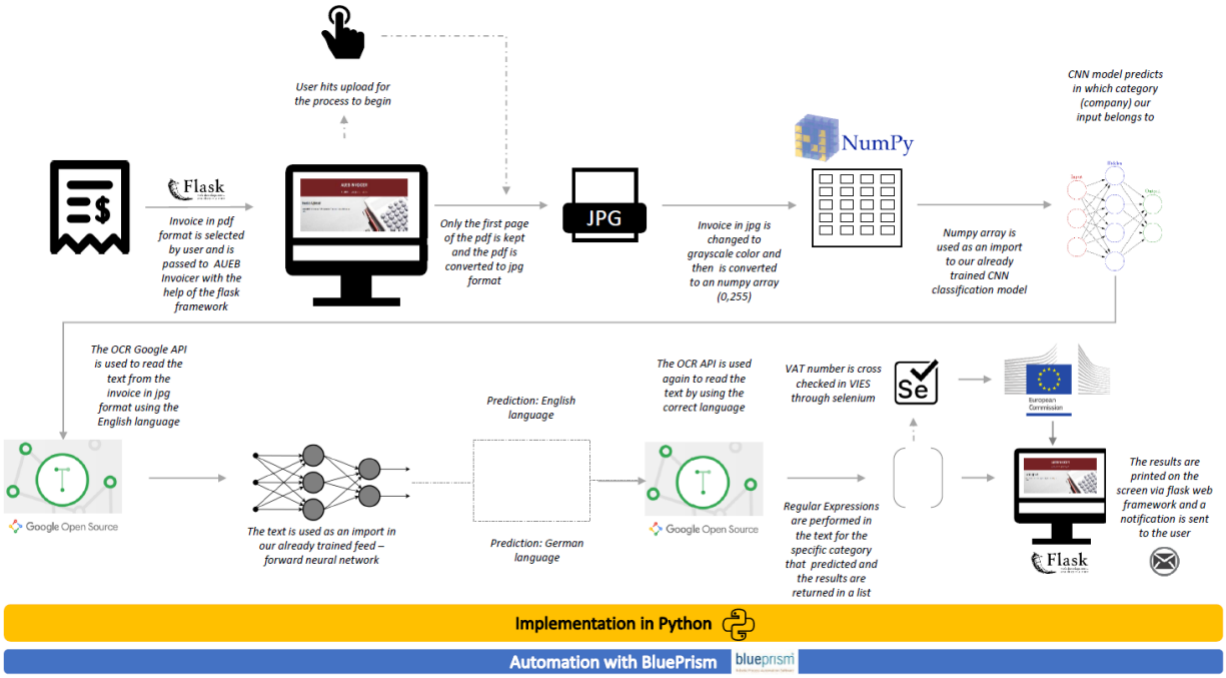


Figure 6: Overall architecture design

As you can see in Figure 6, we describe the whole procedure and what is actually "happening" behind the scenes. In the next chapter, we will describe how we processed our data and how we created our two neural network models.

Last but not least, we have to mention that in the industry there are some applications that perform OCR and text processing (optical character recognition) in documents, but to our knowledge and after a research around the web, there isn't any applications that performs both OCR and text processing on large documents that contain in many parts useful information. There are some solutions that are being used but they refer to the recognition of the serial numbers of products or the number of our credit cards, which is doable considering that in a credit card the card number is located at a certain point, either is an Alpha credit card, or an NBG one for example.

Data

The data that were used in order to train our models, were invoices found in repositories in Github and Kaggle (https://github.com/invoice-x/invoice2data and https://www.kaggle.com/jenswalter/receipts), but also personal invoices. Furthermore, for the language model we have used two text files that included all of the words in English and in German language.

As far as the invoices are concerned, they are coming from various sources, like telecommunication companies, hotels, retail, but there were also some that we have constructed by ourselves. In the neural network model, which recognizes in which company the invoice belongs to, we have used five types of invoices in order to train our model.

Those invoices, were either digital or scanned.

As far as the words are concerned, you can see their distribution below:

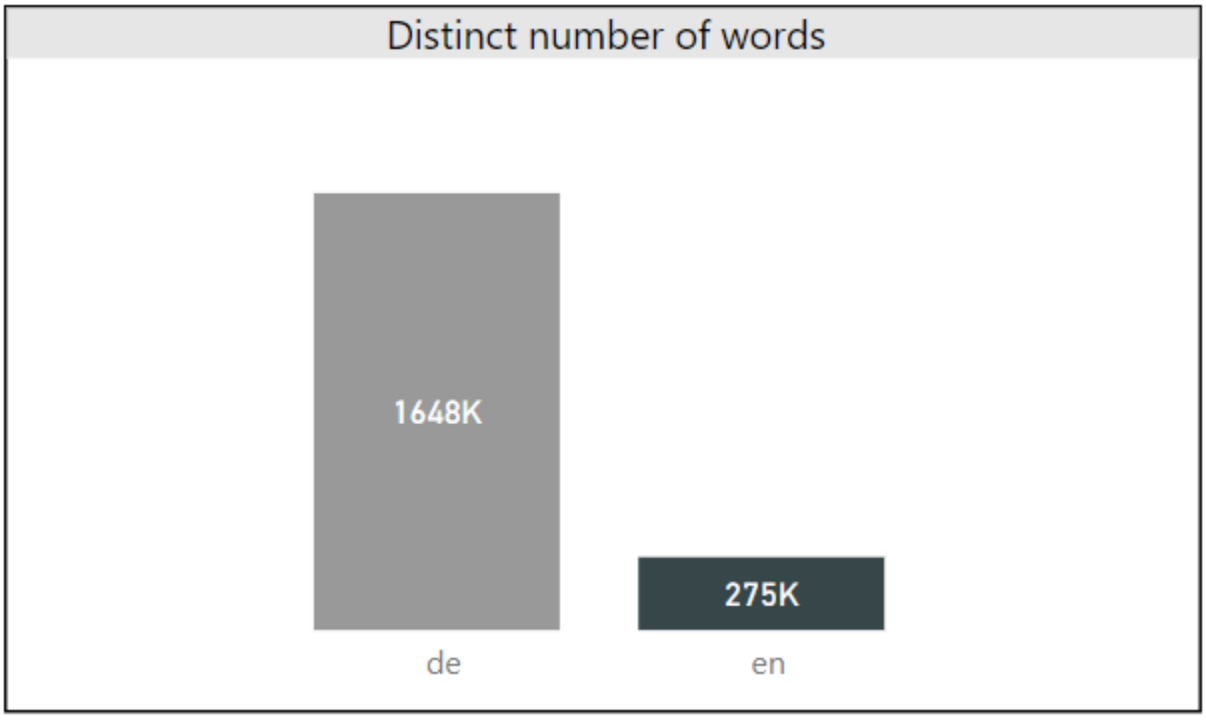


Figure 7: Texts overview

We tried to be consistent on how many invoices will be used for the training, so we decided to keep more or less the same number of invoices for all the five types we had. In that way, we succeeded of not be biased.

For the words, we took a sample of 150,000 words for each language, so that they can be representative of the whole.

The only preprocessing that was performed was converting the pdf format that the invoices were in jpg format and converting to lowercase all the words in the two text files.

Methodology

Two models were used in our application:

1. Convolutional Neural Networks for image classification
2. Feed forward Neural Networks for language prediction

The first model was used in order to predict in which company/vendor the imported invoice belonged to and use the corresponding invoice template. This prediction will be later used in order to know which regular expressions will be used in this invoice, in order to extract the elements that are needed.

For this to happen the neural network model need as an input not an image but an array. Each image for our computers is an array of pixels from 0 (absolute black) to 255 (absolute white) as you can see also below:

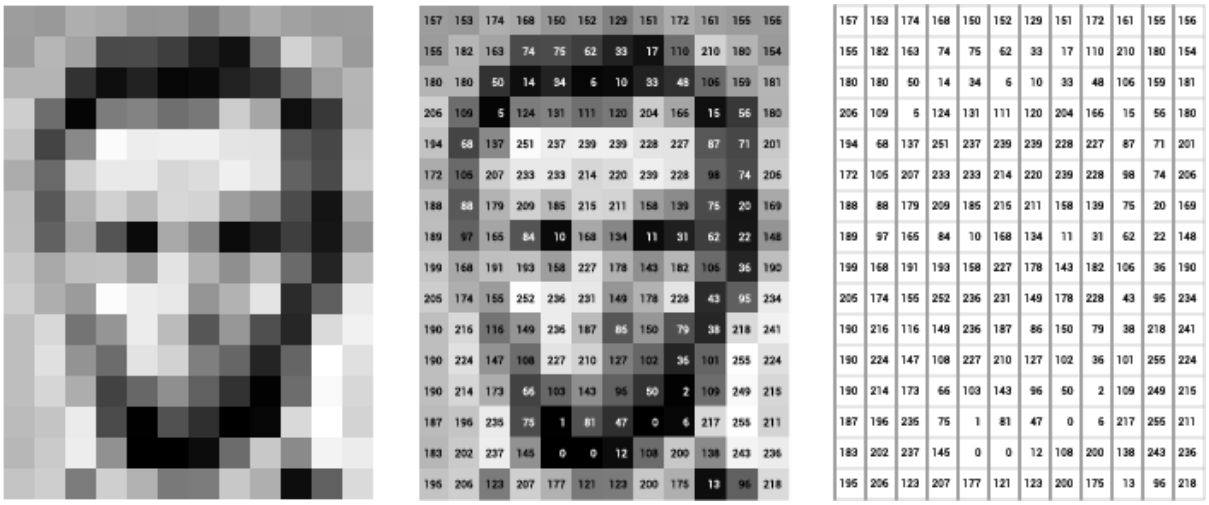


Figure 8: Image to array

We first grayscale our image, meaning make it only black and white, so that we can only have on array to manipulate, then we reshape the image to 50 x 50.

Also, in order to label our images by company/vendor, we assign to each vendor/company a vector, having the form [1,0,0,0,0] for the first one, [0,1,0,0,0] for the second one etc. We have used invoices from five vendors.

Having converted the image to array, we then import to the model.

Our neural network has an input layer, next we have 8 layers of convolution and pooling, then we add a fully connected layer and then the output layer. The activation functions that were used were relu and softmax. Also, we have adam optimizer, and a learning rate of 0.003.

You can see how the neural network is constructed below:

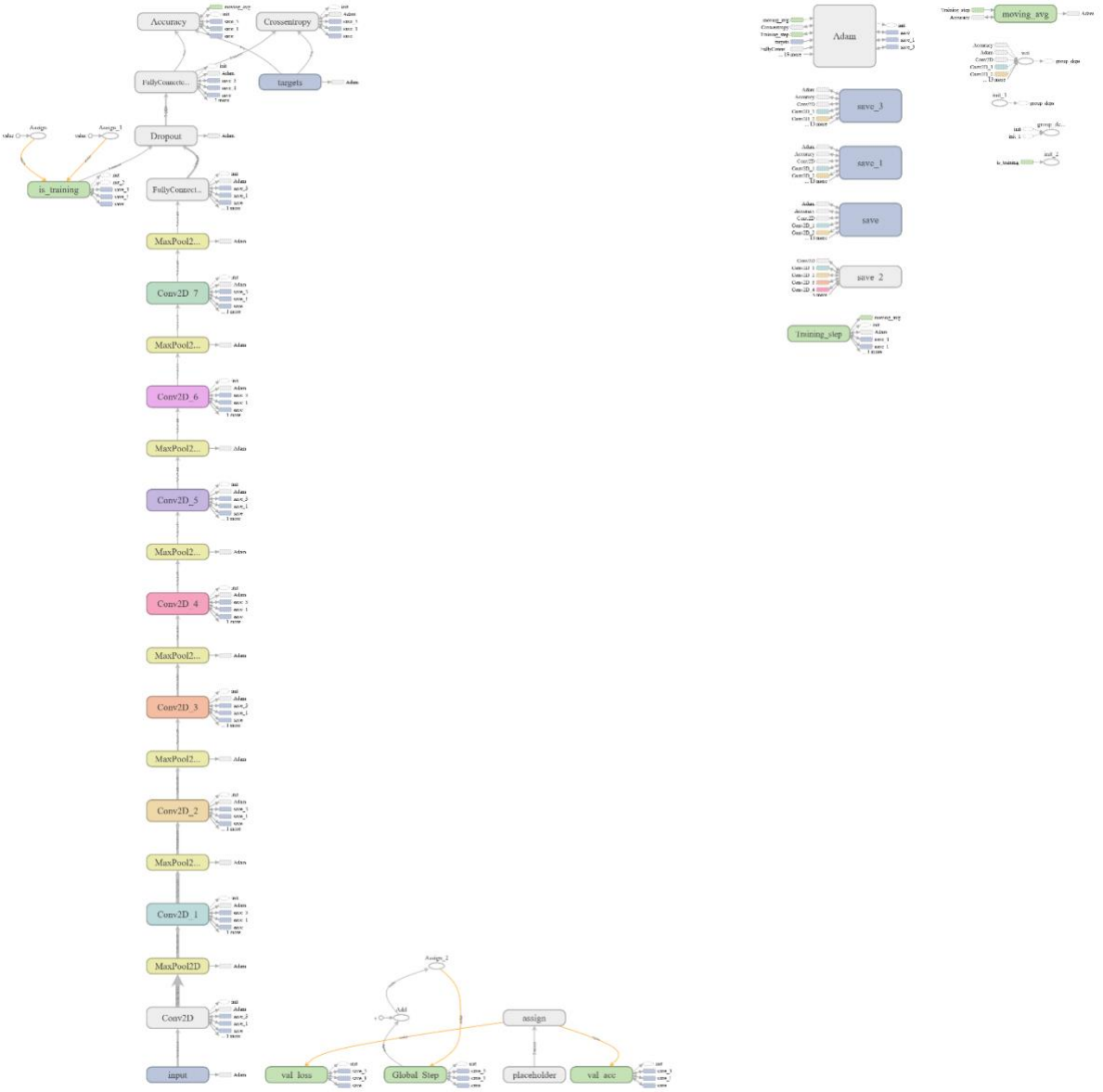


Figure 9: Convolutional Neural Network model

As far as the evaluation is concerned, we kept only the accuracy measure, and we trained our model for 100 epochs.

In order to conclude to the above, we had to test our model. We tested and compared our model, with the same model but with different parameters, such as learning rate, epochs and a smaller number of invoices as an input. We have to say that the model predicts quite well all the labels.

As far as the language model is concerned we have used feed forward neural networks.

Let's first explain how we transformed the data. In order to train/test our data we used the labeling of each word (English or German). So we have

- [1,0] for English
- [0,1] for German

After that we converted each letter of the words we had into numbers as can be seen below:

a=10000000000000000000000000
b=01000000000000000000000000
c=00100000000000000000000000
…
z=00000000000000000000000001

Figure 10: Letters to numbers

Then, for each word we bind the numbers together and we complete the remaining words with zeros. This is highly important, because we set the max letters of a word to be 18. For example, the word "hello" can be seen below:

Figure 11: Representation of the word "hello"

We can see how another word is represented into a python array:

	0	1	2	3	4	5	...	465	466	467	468	469	470
reimagine	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

Figure 12: Python array

We then load our data, and split it into two arrays, inputs and labels. Inputs just contains the 468 element long arrays representing all of the words, and labels contains the 2 element long arrays representing the language tags. We split these arrays into training and testing data, with 15% of the data being used for testing

We will be using a feed-forward network with four hidden layers, each with 200, 150, 100, 100 nodes respectively. Each layer uses a sigmoid activation function except the final layer which uses softmax, as this function allows us to assign a probability to each language, as the sum of the output layer values will be one. We will use the Adam optimiser, sgd optimizer, an adaptive learning rate, gradient descent optimiser, and we will use the binary cross entropy loss function. The only metric we will worry about for this project is the accuracy of the model.

Here is the output of our model in graphical form through Tensorboard:

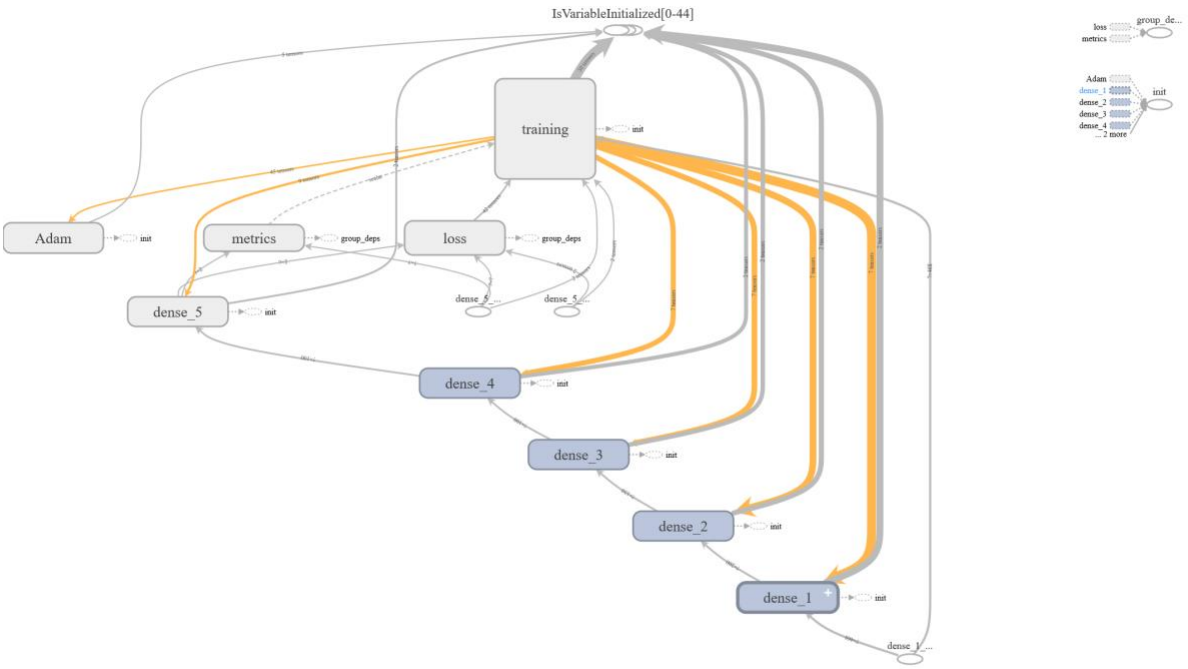


Figure 13: Feed Forward Neural Networks model

As a last step, the model gives as an output a percentage in which language each word belongs to.

```
en: 94.49%
de: 5.51%
en: 2.76%
de: 97.24%
en: 97.65%
de: 2.35%
en: 99.15%
de: 0.85%
en: 99.21%
de: 0.79%
en: 33.82%
de: 66.18%
en: 0.01%
de: 99.99%
```

Figure 14: Probability (%) of language

We then, for each language sum up the probabilities and conclude depending on the result in which language the whole text belongs to.

In order to conclude to the above, we had to test our model. We tested and compared our model, with the same model but with different parameters, such as learning rate, epochs and a smaller number of words as an input. We have to say that the model predicts quite well the language inside the text.

Results

After explaining how the two models were developed, we are ready to present the results. As we mentioned above the only measures that were used, were accuracy and loss.

So for the first model that refers to the classification of invoices we have:

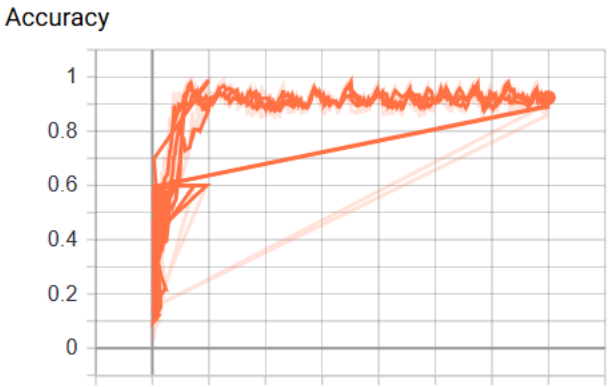


Figure 15: Validation Accuracy - 92% (training 100 epochs - 458 images)

Along with the accuracy we should evaluate the loss function, i.e. this is a measure that can help us understand if we minimized the errors in our model.

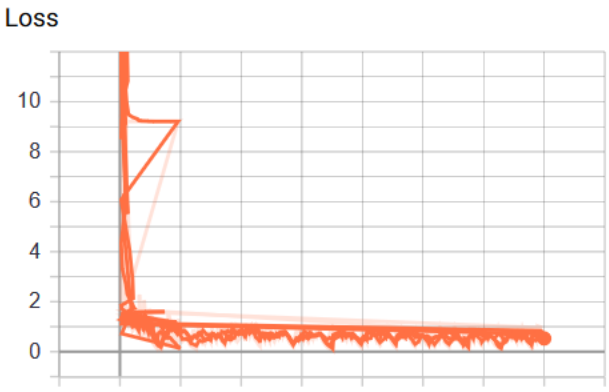


Figure 16: Validation Loss – 5%

We also compared our model by running it for different epochs, train and test data and more or less hidden layers. Below you can see the progress in accuracy as we try different techniques. This graph is feasible because Tensorboard keeps the history of the runs we have made.

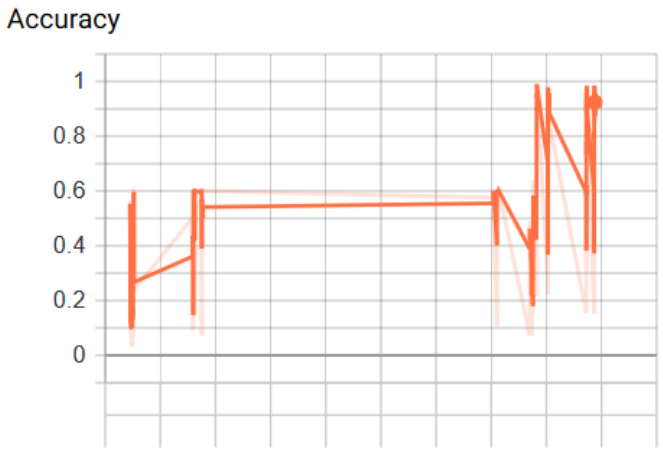


Figure 17: Validation accuracy progress

The problem was located to the number of invoices we had at the beginning for each vendor/company, when we increased the number accuracy improved significantly.

The training of this model did not take enough time, due to the small number of images we had. It took approximately 10 minutes.

Let's see the loss as well:

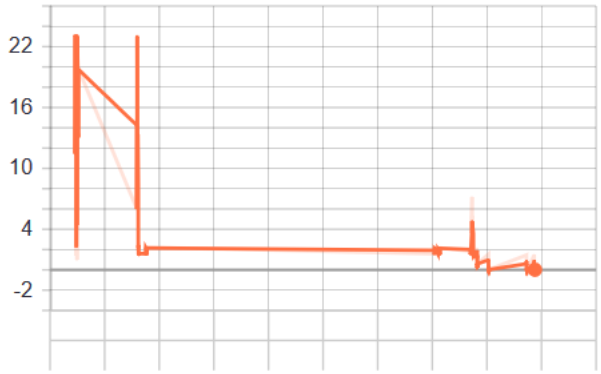


Figure 18: Validation loss progress

As far as the language model is concerned, we have:

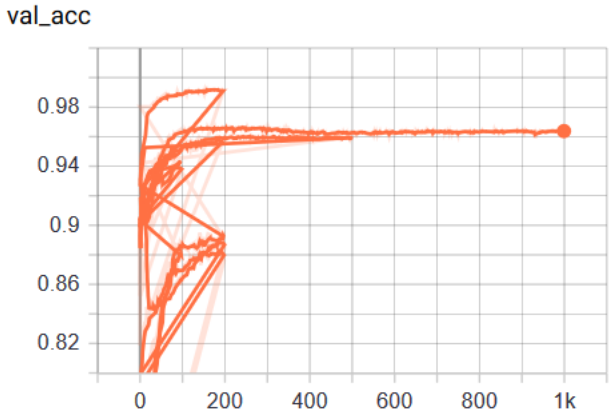


Figure 19: Validation accuracy - 96%

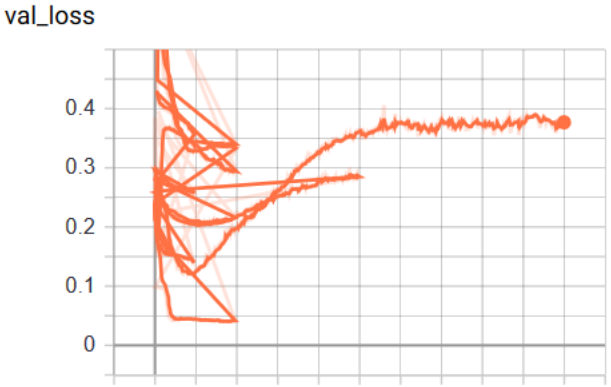


Figure 20: Loss - 38%

As far as the comparison is concerned, as above, we run the model for different epochs, train and test data and more or less hidden layers. Below you can see the progress in accuracy as we try different techniques. This graph is feasible because Tensorboard keeps the history of the runs we have made.

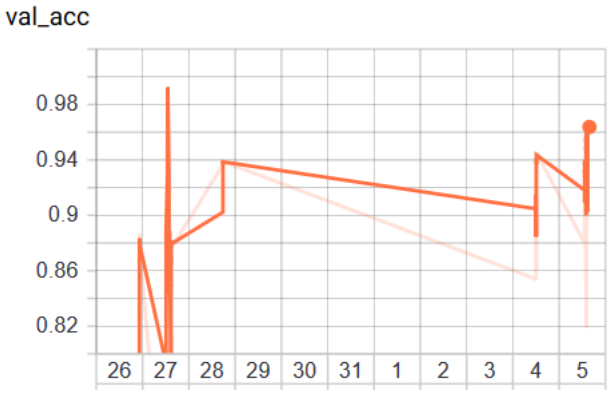


Figure 21: Accuracy progress (in days)

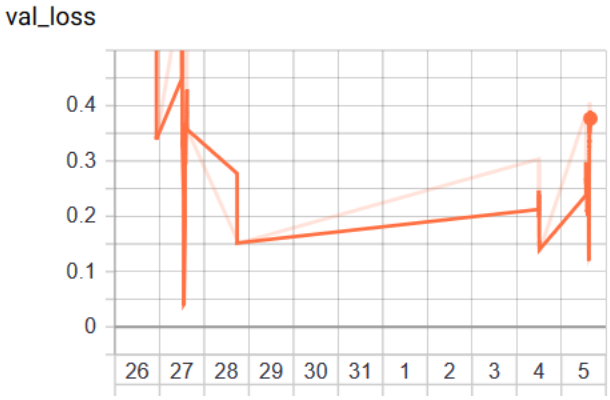


Figure 22: Loss progress (in days)

The main problem here was that the sample we initially took was too small. Also, it needed more epochs of training. We resulted that an efficient number of epochs are 1000.

The training of this model has taken enough time, because we had a fairly large number of words (~ 300,000). We run the model for 1,000 epochs. The training's duration was approximately 2 hours.

All the above were run in *JetBrain's PYcharm* in a dell laptop with 16GB of memory, in a 64bit system and an i5 – core. *Python 3.7.3* was used.

Robotic Process Automation (RPA)

1.Functional Requirements

In order to minimize the processing time of the entire process, it is designed to fully automate the process using an RPA solution, so that a robot implements the steps that a human would carry out in order to set up and trigger the OCR procedure.
The basis for initiating the process is the identification of e-mails with attached invoices. Specifically, the robot would open the Outlook and would store in a collection all the data of the unread mails in the subject of which there is the word "Invoice". As next step of the process, the robot saves the attached invoices in a predefined folder, the path of which is stored as environmental variable in Blue Prism with the name "C:\Users\akyrgiou\Desktop\Invoices". Then, the robot is redirected to the screen of the web base application "AUEB Invoicers" where it chooses one invoice to upload in order to trigger the OCR procedure.
At the beginning of the process the robot sends an email to the responsible personnel to inform them that the process is initiated. When the process is completed the relevant email is sent

accordingly in order to inform for the successfully termination of the process. In case of failure the email consists of the exception message that describes the error.

2.Special Standards

2.1 Development Standards

Code: BluePrism Flows & .NET (Visual Basic, C#)
Graphical User Interface: BluePrism GUI
Application: Navigation to the application “AUEB Invoicers”

2.2 Configuration Management

In a real business case, the development would take place in a properly configured development environment (DEV) whereas the user acceptance test would take place accordingly in a QA environment. For the needs of this project the DEV and QA environments are the same.
Finally, the source code is maintained in the database of each blue prism software environment.

3.System Architecture

3.1 Architectural Design

The platform of Blue Prism (BP) was used in order to develop the code for the implementation of RPA. The standard installation of Blue Prism software requires a minimum set of four computing (virtual or physical) machines including:

- Application Server
- Interactive Client (Human)
- Runtime Resource (Robot)
- SQL Server

In a real business case three different environments should be set up: the development, the test and the production. In addition, the aforementioned components should be installed in infrastructures:

1. Admin Machines which are the physical PCs from which the operation can access the Blue Prism Interactive Client (Development Studio, Controller) via Remote Desktop
2. The Virtual Machines, where the BP Interactive Clients are installed
3. The Virtual Machines, where the BP Runtime Resources (the robot) perform the work
4. The Blue Prism’s virtual application server
5. The Database server
6. The Production applications accessed by Robots

For the need of our project, all the infrastructures are set up in the same PC.

3.2 System Environments

Interactive Client & Runtime Resource

Hardware

- Intel Processor (2 Processors)
- 8GB RAM
- 150GB free disk space (after install of Operating System and standard software)

Software

- Windows 10 Enterprise 64-bit version.
- .NET Framework 4.7
- Outlook (32-bit Outlook 2003 or later)
- Microsoft Office
- Firewall adjusted for the communication with the Application server

Application Server

Hardware

- Intel Xeon Processor (2 Processors)
- 8GB RAM
- 80GB free disk space after install of OS and standard software

Software

- Windows Server 2016. 64-bit version
- .NET Framework 4.7
- Firewall adjusted for the communication with the VDs

Database Server

Hardware

- Intel Quad Xeon Processor (2 processors)
- 8GB RAM

Software

- SQL Server 2016 64-bit version.
- Case insensitive, 1252 code page SQL Collation
- Non-clustered or clustered architecture (includes support for AlwaysOn Availability Groups)
- Allocate 10GB Data File per connected Runtime Resource (min 100GB)
- Allocate 5GB Data Log File per connected Runtime Resource (min 50GB)
- High performance disk array

3.3 Security Architecture

In General, BP consists of nine roles with different rights, as described in the table below:

Blue Prism Roles	*Description*
System Administrator	System administrator. He has no restrictions.
Process Administrator	He has access to BP execution sessions, queues and logs.

Schedule Manager	He has full access to the Control Room and process scheduling.
Alert Subscriber	He can trigger alerts within the Interactive Client's BluePrism.
Developer	Developer Can create, edit, delete and execute all processes and VBOs.
Tester	He can execute processes and VBOs. He, also has access to execution sessions.
Release Manager	He can import, export and copy processes and VBOs between different environments.
Role Manager	He defines the roles of Users.
Auditor	He has read only rights.

Table 1 - Blue Prism Roles

4.Blue Prism Development

The implementation of RPA techniques is implemented by using three layers: Processes, Business Objects (VBOs) and Actions. Each process consists of a main page and many other sub-pages depending the case. The main page must be kept simple because it is easier to understand what the process does just by looking at it. Main page is the one that initiates the process and calls the other sub-pages (work pages) which are used for navigating and updating the systems that the human would use in order to complete the process manually.
The work pages call the VBOs, a VBO is the instrument that a process uses to control any application and provide a set of simple functions orchestrated into a complex sequence based on the case and the business requirements. The object layer is for application logic only and consists of the Actions that should be small, generic and re-useable.
So, a process consists of one main page which calls many sub-pages and each sub-page calls one or more VBOs. Each VBO consists of two or more pages, the first for initiating reasons and the rest for implementing the necessary actions. It is preferable to keep a VBO's page simple without many actions and create more pages so that the VBO is more efficient.
Last but not least, it is important the implementation of exception handling so that the user could understand in case of failure what error occurred and in which step of the process. The exception handling includes the following:

- Retry exception handling logic on sub-pages.
- An Exception Block on the Main Page around sub-pages that interface with systems.
- A Mark Item as Exception sub-page that terminates the process if the same system issue occurs for concurrent work items.

Robustness: Sub-Page Retry Loops
If an exception occurs on a sub-page it could be a one-off issue (i.e. a network timeout) that would be fixed if the process flow simply tried again. All 'work' Sub-Pages should do the following:

1. Catch any exceptions with a Recover stage.
2. Evaluate the exception with a Retry? decision

3. If there has been less than 3 attempts and the exception is a type you want to retry, loop around and try again.
4. If trying again, perform actions to 'tidy up' the system being used, in the flow on the right we simply restart it but you may need to re-navigate back to the correct system state.
5. If not trying again, 'throw' or 'preserve' the exception up to the Main Page.

Robustness: Main Page Exception Block
If an exception 'bubbles up' to your Main Page from a Sub-Page the following steps take place:

1. A block around the main work interface sub-pages, with a recover stage.
2. A Mark Exception action that will contain the case exception handling logic (marking the item as an exception).

Robustness: Consecutive Exceptions
If the same exception occurs for every Work Queue item a process attempts to work, it is best to terminate the process so that a Controller can investigate the issue.

Database Construction

In a real business case scenario, from AUEB Invoicer we would gather all the appropriate information and save them in a database having as invoice status pending by default. Then any appropriate change, further processing, update of the invoice status as well as checking procedure would take place through an ERP system. The final reporting and monitoring phase would be implemented through Power BI. The suggested workflow is the following.

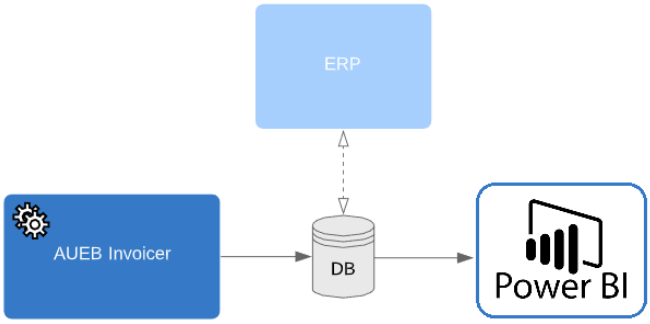


Figure 23 Database/ERP workflow

Further, we designed the database in SQL where all the information would be gathered, organized and easily managed. The Entity-Relationship Diagram (ERD) of our database can be found at the next figure.
We have implemented a star schema having the Invoices table as our main table where all the transactions are kept through AUEB Invoicer and ERP system.
The rest tables containing the dimensions are:

- The calendar table where we keep the dimension of time and is joined with the main table with the Invoice Date.
- Invoice Status where we keep the dimension of the status of our invoices e.g. pending, paid or rejected, and joined with unique StatusID
- The currency table where we keep the dimension of amount's currency e.g. euro, dollar or rupee and joined with the main table with unique CurrencyID.
- Business Partners table where we keep the dimension of our business partners and some further details as their country/region, joined with the main table with the unique BusinessPartnerID
- And the final two tables USD rates and Rupee rates that contain the daily currency rates EUR/USD and EUR/RUP. Those tables are necessary in order to make the conversion of our invoices (to euro) with the actual rate of the invoiced date. Those rates are imported through the European Central Bank site.
  https://www.ecb.europa.eu/home/html/index.en.html

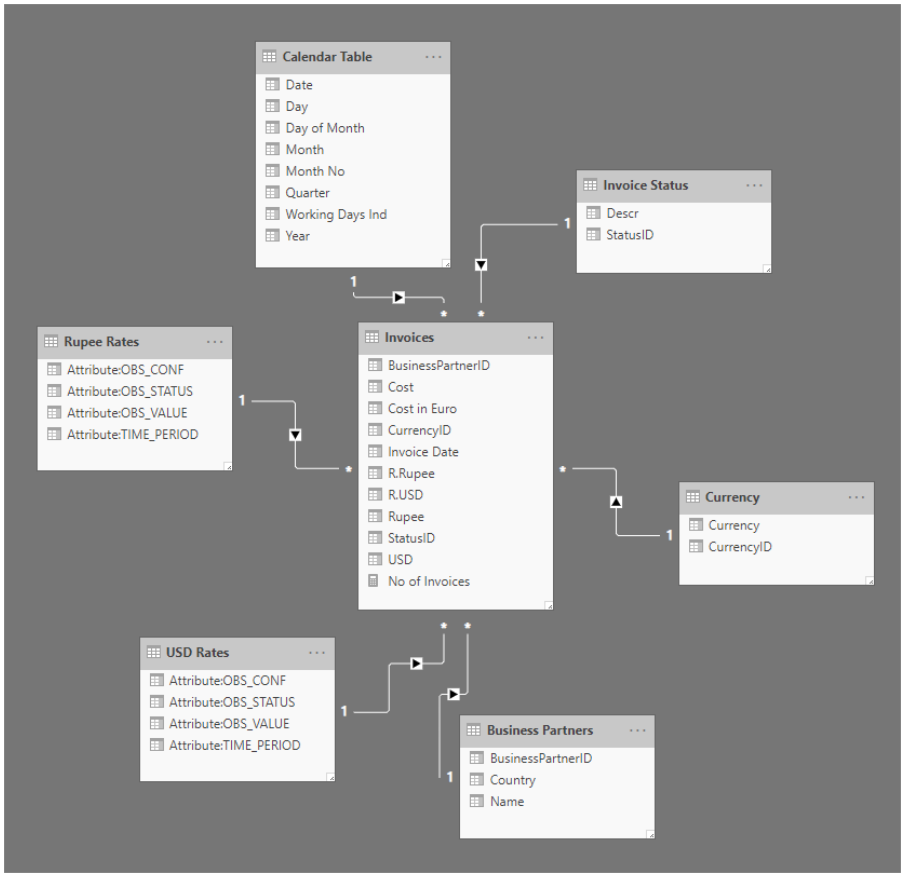


Figure 24 Entity-Relationship Diagram

Monitoring Tool

Now that the database is up and running, we imported some dummy data in order to test it and proceed to the monitor/reporting phase.

After connecting our SQL database with power bi, we constructed this report where we can easily find useful information and insights regarding company's expenditures. Trend analysis, vendors analysis as well as paying lag are only some of the monitoring reports that can add value to the company.

Future Work

In light of the above, some challenging upgrades to the recent assignment/project would be to include in our model the capability of understanding and capture: any invoice format changes, as well as any new vendor that we haven't done business with them before and re-run the process as already mentioned.
Moreover, another useful add-on would be to mass upload many invoices into batches and again extract, transform and load every valuable information from them in an effortless and rapid way.

References/Bibliography

1. *https://medium.com/coinmonks/language-prediction-using-deep-neural-networks-42eb131444a5*
2. *https://stackoverflow.com/questions/17759860/python-2-smtpserverdisconnected-connection-unexpectedly-closed*
3. *https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/*
4. *https://medium.com/coinmonks/language-prediction-using-deep-neural-networks-42eb131444a5*
5. *https://www.youtube.com/watch?v=gT4F3HGYXf4&t=634s*
6. *https://www.information-age.com/optical-character-recognition-tools-ocr-ai-123479324/*
7. *https://www.ecmconnection.com/doc/optical-character-recognition-the-mature-tech-0001*
8. *https://www.ecb.europa.eu/home/html/index.en.html*

Appendix

List of Business Objects

VBO Name	***VBO Description***	***Action***	***Description***
Utility-Environment	VBO for interacting with the environment	Get_Robotic_Account_Name	Action that retrieves the user name.
Utility-Environment	VBO for interacting with the environment	Get_Robotic_Machine_Name	Action that retrieves the name of the PC.
MS Outlook Email VBO	VBO for interacting with Microsoft Outlook email functions.	Send_Email	Action that creates and sends an email by providing the required variables (To, Subject, Body)
MS Outlook Email VBO	VBO for interacting with Microsoft Outlook email functions.	MS Outlook Email VBO::Get Received Items (Basic)	Action that lists Inbox unread items and store the data in a collection name *"col_Items"*
Utility - Collection Manipulation	VBO for filtering a collection using the given query each time.	Utility - Collection Manipulation:: Filter Collection	Action that filters the col_items based on subject. If there are emails with the word Invoice in their subject, this action updates the collection *"col_Items"* with them.
MS Outlook Email VBO	VBO for interacting with Microsoft Outlook email functions.	MS Outlook Email VBO::Save Attachment	Action that uses the Entry ID from the aforementioned collection as a unique identifier to determine from which email it will download the attached invoice. The folder in which the attachment will be downloaded is the *"C:\Users\akyrgiou\Desktop\Invoices".*
Collections	VBO that runs in the background and manipulates the collections	Count Rows	Action that counts how many rows the collection has.
Collections	VBO that runs in the background and	Add_Email_Data_Row	Action that adds an empty row in the collection *"col_BusinessEmailData"*

	manipulates the collections		that stores the data for the email to be sent to Business in case of failure.
MS Outlook Email VBO	VBO for interacting with Microsoft Outlook email functions.	Send_Email	Action that creates and sends an email by providing the required variables from the *"col_BusinessEmailData".*

Table 2 - List of Used VBOs
**The VBOs are dynamic in order to be reusable based on the input and output parameters as defined in the process*

Invoice Alert email