

M2 WRITEBACK EXTENSION USE CASE

Text classification with human-in-the-loop machine learning.

Vahe Shelunts (1850563)

Quynh Tran (569677)

Elias Brummund (519999)

M2. technology & project consulting GmbH
HWR Berlin - Enterprise Architectures of Big Data

Table of Contents

TABLE OF FIGURES.....	II
1. OVERVIEW	1
1.1. CONCEPT IDEA	1
1.2. SOLUTION ARCHITECTURE.....	2
2. THE DATA	5
2.1. ABOUT THE DATA.....	5
2.2. SETTING UP THE DATABASE	6
2.3. FILLING DATABASE WITH DATA.....	9
2.4. DATA PREPROCESSING.....	10
3. THE ALGORITHM	11
3.1. SUPERVISED TEXT CLASSIFICATION	11
3.2. SERVER DEPLOYMENT & AUTOMATION	12
4. M2 WRITEBACK EXTENSION	16
4.1. FUNCTIONALITY	17
4.2. IMPLEMENTATION	20
4.3. HUMAN-IN-THE-LOOP MACHINE LEARNING	22
5. OUTLOOK.....	24
REFERENCES	25

Table of Figures

Figure 1: Solution Architecture	4
Figure 2: 20 Newsgroup Organization.....	5
Figure 3: Sample text from 20Newsgroup dataset	6
Figure 4: Docker compose YML.....	8
Figure 5: SSH authorized keys.....	13
Figure 6: Crontab.....	15
Figure 7: Writeback tool Form elements.....	18
Figure 8: Before and After – the possibilities of Tableau Writeback Extension.....	19
Figure 9: Implementation overview of the Writeback tool.....	20
Figure 10: SQL query integrated in the Writeback tool.....	20
Figure 11: Writeback form designed for 20NewsGroup email dataset	21
Figure 12: Extract of the database output	22

1. Overview

1.1. Concept Idea

M2. technology & project consulting GmbH (short: M2) is a Berlin based company founded in 2009 that offers NextGen Business Intelligence consulting and solutions. The key aspect of the work of M2 is to unleash the potential of their clients' data. As the first German partner of Tableau software and Tableau DACH Partner of the year 2020, M2 also does inhouse development of Tableau Extensions.¹ The most recent is the Writeback Extension, which was the crucial point of our project. The Writeback Extension offers the possibility to write custom data to the data source from the dashboard.²

Since the Writeback Extension is rather new, the current demo version just shows basic functionalities of the tool and how it works. In order to market it better and to show how to integrate the extension into real environments, it was our task to come up with an idea for a machine learning use case with a business context and to realize this idea. The purpose of the project is to show the result to clients as a second demo that outlines how they can benefit from adding the writeback extension to their maybe already existing business intelligence architecture or how to set up an architecture they can benefit from.

¹ (M2. technology & project consulting GmbH, n.d.)

² (M2. technology & project consulting GmbH, n.d.)

Therefore, our idea was to use machine learning in the business context of email classification and to show how the correction of misclassifications with the writeback extension enables human-in-the-loop machine learning. As the name already implies, human knowledge and machine intelligence will be combined to increase accuracy.³ This is done by using the classified data, in our case emails, to train the classification algorithm so it can learn from the new data. In order to not train misclassified data, which can lead to bias and future misclassifications, the Writeback Extension is used to correct these to ensure that only correct classifications are trained. With this technique, one can use the new input of data to quickly grow the training dataset, which can lead to an improvement of the model performance in a short time.

Throughout the project a lot of scripts from a repository are mentioned. Please find the repository attached to the upload of this assignment in moodle.

1.2. Solution Architecture

As can be seen in Figure 1 below, to realize our project idea we used a variety of different software and service providers. Continuous lines indicate that a solution architecture component is stored or hosted on another solution architecture component. Dotted lines indicate communication and exchange of data between components. The arrows on the dotted lines indicate in which direction data flows.

We (Vahe, Quynh and Elias) are building the top of the architecture, because we're the owners and developers of this solution. There is a flow of data between us and the architecture, since we integrated and accessed data to and from the solution

³ (Monarch, 2021)

architecture, when building and using it. PyCharm with GitHub integration was used for development and version control.

The basis for the left side of our solution architecture is a virtual private server (VPS) hosted by Hetzner in order to host Python code and a MySQL database via Docker. As an operating system on the Hetzner VPS we've chosen Ubuntu since it is open source and has a wide range of functionalities as a server operating system. For example, Ubuntu's utility crontab proved itself as an important component to our project result later on. The dotted line with arrows into both directions between Python and MySQL means that the Python code grabs data from the MySQL database to process it and then writes the result to the database. This is done via the MySQL Python connector library.

A similar relationship can be seen between Tableau and MySQL. A dotted line between Tableau and MySQL with an arrow pointing to Tableau is normal since it is Tableau's purpose to receive data and to visualize it. The key to our project is the arrow pointing back from Tableau to MySQL, since this is realized only by the M2 Writeback Extension, which allows to write custom data into a database from Tableau. This part enables human-in-the-loop machine learning with Tableau and is the highlighted component in our project.

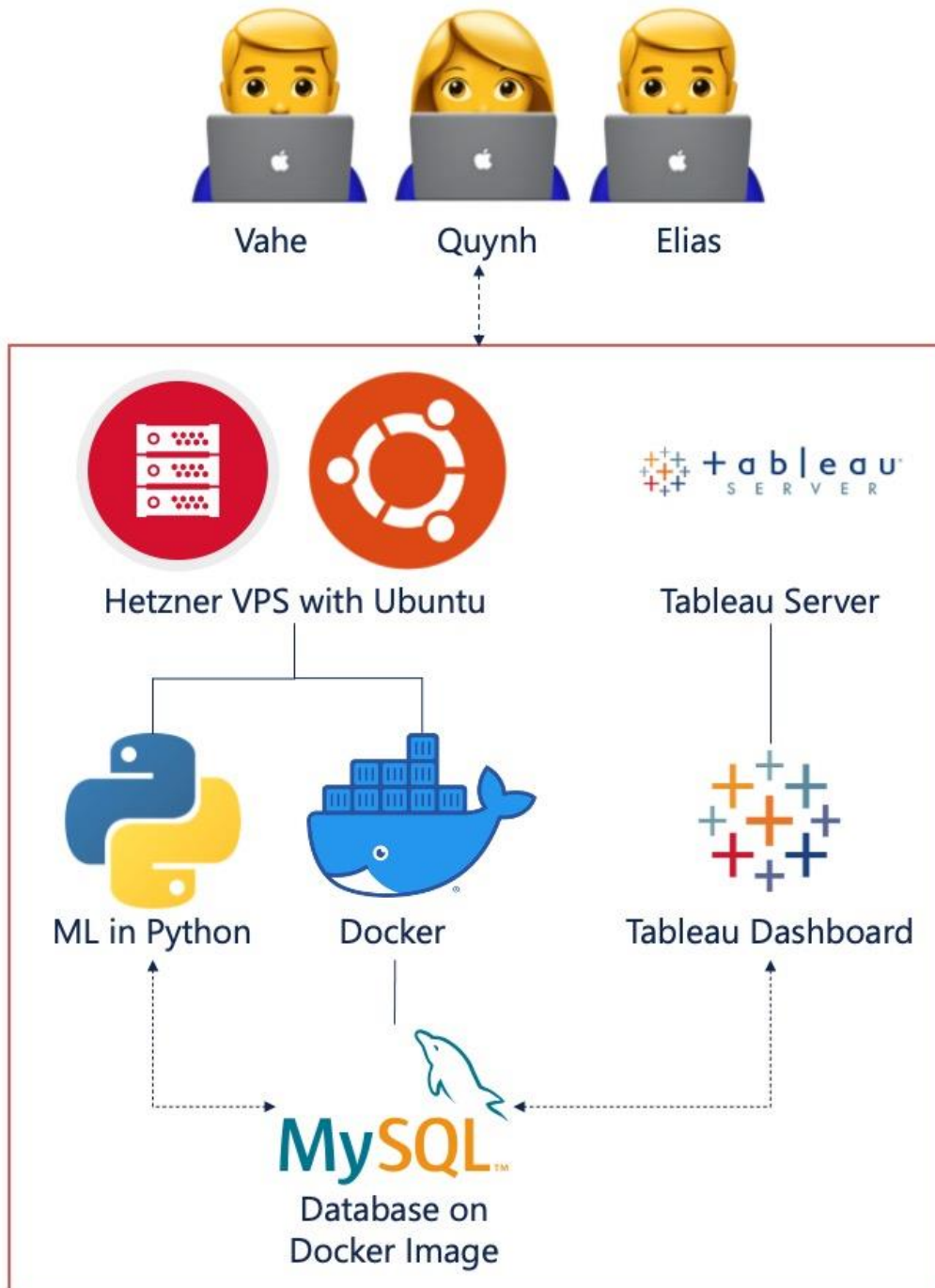


Figure 1: Solution Architecture

2. The Data

The foundation of our use case, in which we want to improve a model's performance with the help of the input via the Writeback Extension, is built up on the 20 newsgroups data set. The following sections explain how the data is structured and preprocessed and the approach we took for setting up the database and finally loading the data into the database to make it accessible for the model.

2.1. About the Data

The 20 Newsgroups data set is a collection of approximately 20000 e-mail texts that are partitioned into 20 different newsgroups. Each of the categories is corresponding to a different topic. However, some of the topics are closely related to each other. The organization of how the categories are structured can be seen in.⁴

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Figure 2: 20 Newsgroup Organization

⁴ (Selva86, n.d.)

An exemplary text for the newsgroup 'sci.electronics' is shown in Figure 3.

Each text consists of the same parts which are separated by line breaks:

1. The sender (e-mail address and name)
2. Subject
3. E-mail content

```
'From: ray@ole.cdac.com (Ray Berry)\nSubject: Re: $25 network\nOrganiza
tion: Cascade Design Automation\nLines: 11\n\nnzjoc01@hou.amoco.com (Jac
k O. Coats) writes:\n\n>The same folks now have out LBL (Little Big Lan
) for $75. I think you\n>get it for $50 if you already own $25 Network
. LBL works with Arcnet,\n>parallel ports, and serial ports in any com
bination for up to 250 or so\n>nodes.\n\n    LBL now offers ethernet su
pport also, although presently it is limited\nto NE1000/NE2000 style bo
ards. LBL owners can get an update for $8.50.\n-- \nRay Berry kb7ht ra
y@ole.cdac.com  rjberry@eskimo.com  73407.3152@compuserve.com\n'
```

Figure 3: Sample text from 20Newsgroup dataset

Since the collection is a popular and well-known data set in the field of machine learning and text classification, it was the main reason why we have chosen it for our use case demonstration.

2.2. Setting up the Database

As the Writeback Extensions only supports relational databases, we've chosen a MySQL database to store our data, because it is open source and provides a good documentation. As described in the foregoing section, the main data in this project are emails. One could argue that non-relational databases like MongoDB would fit better, because handling textual data takes a lot of storage and MongoDB is considered to be faster when processing large amounts of data and easier to scale up in comparison to

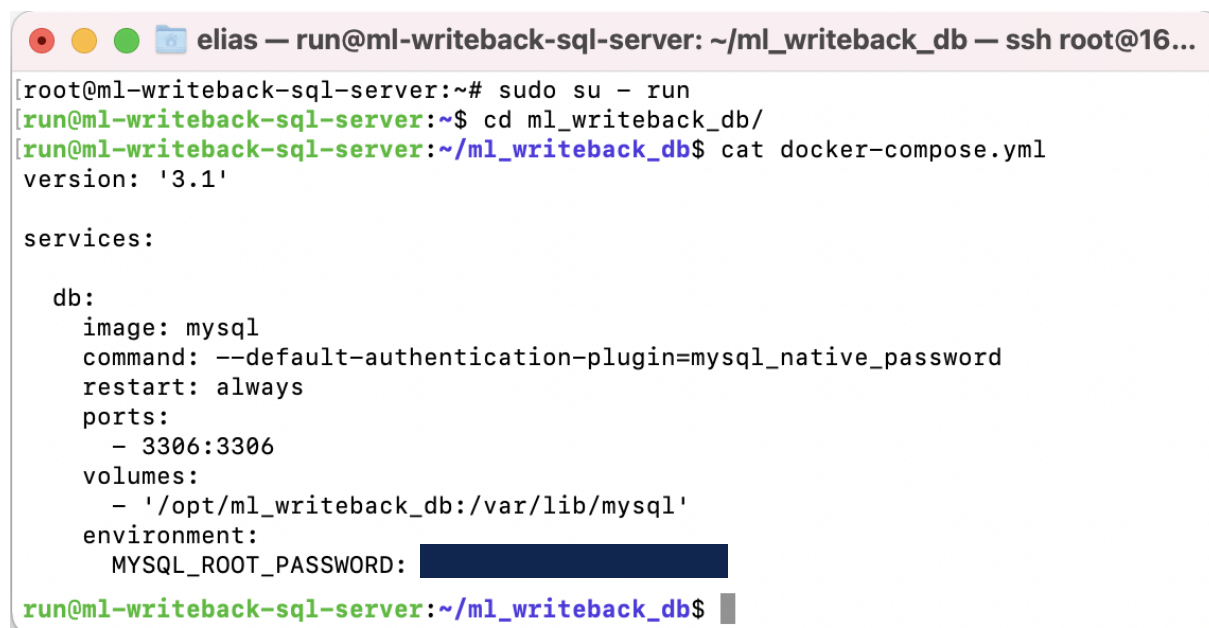
MySQL.⁵ However, regardless of the fact that MongoDB isn't supported by the Writeback Extension, a relational database in the form of MySQL fits the purpose of our project better, since we depend on clearly structured data and consistency. In order to save classification results and histories to track model classifications and classification changes by users a structured database is more suitable. Moreover, it is important to maintain consistency, because in document classification use cases the documents are usually of high importance. Therefore, it should be ensured that all documents received are listed in the database and directly labelled in order to show up for the respective department and not get lost as not labelled.

The use of Docker to install the MySQL database was chosen to create an isolated environment for the database to ensure it is working properly, to easily transfer it to another server or production environment after the project is finished (if needed) and to easily scale in case it'll be used for further development after the project. But before installing the MySQL database, Docker Engine and Docker Compose needed to be installed on Ubuntu. The official Docker Engine installation guide was followed, and Docker Engine was installed from a Docker repository as it was recommended in the guide.⁶ First a repository was set up. Apt packages like transport-https, ca-certifications, curl, gnupg and lsb-release were installed in order to use a repository over HTTPS. After this was done, Docker's official GPG key was added, and a stable repository was set up. Then, just a simple apt-get install command was needed to install Docker. Since Docker Hub offers an official MySQL image, the installation of the MySQL Database was also rather easy and without complications. Afterwards the official Docker Compose

⁵ (MongoDB, 2021)

⁶ (Docker, 2021)

installation guide was followed and a stable version was downloaded before executable permission to the binary was applied.⁷

A screenshot of a terminal window with a pink title bar. The title bar text is "elias — run@ml-writeback-sql-server: ~/ml_writeback_db — ssh root@16...". The terminal shows a sequence of commands and their output. First, "sudo su - run" is executed. Then, "cd ml_writeback_db/" is executed. Finally, "cat docker-compose.yml" is executed, displaying the content of the file. The content includes a version field set to '3.1' and a services section with a 'db' service. The 'db' service specifies the 'mysql' image, a command for authentication, a restart policy of 'always', port 3306, and a volume mount. An environment variable for the MySQL root password is also defined, with the password value redacted by a black box. The prompt at the bottom is "run@ml-writeback-sql-server:~/ml_writeback_db\$".

```
[root@ml-writeback-sql-server:~# sudo su - run ]
[run@ml-writeback-sql-server:~$ cd ml_writeback_db/ ]
[run@ml-writeback-sql-server:~/ml_writeback_db$ cat docker-compose.yml ]
version: '3.1'

services:

  db:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    ports:
      - 3306:3306
    volumes:
      - '/opt/ml_writeback_db:/var/lib/mysql'
    environment:
      MYSQL_ROOT_PASSWORD: [REDACTED]
run@ml-writeback-sql-server:~/ml_writeback_db$
```

Figure 4: Docker compose YML

Now that Docker Compose was installed, the YML file was written. As seen in the figure above, the image of the service "db" was specified as "mysql", which automatically grabs the official MySQL image from Docker Hub. The executed command in the second line of the service "db" defines that authentication to the database is done via password. The password is then defined as an environment variable further down. "Restart" is set to always, which means that the container is automatically started in case the machine, docker or the container itself are restarted for whatever reason. This saves time in case of an outage of the server. The port tags define the host and container port to be 3306. With the volume tag a folder from the host is mounted to

⁷ (Docker Docs, 2021)

the container. This ensures that the data saved in the MySQL database is not lost when the container is restarted or deleted for whatever reason. After executing "docker-compose up" the database was up and running.

2.3. Filling Database with Data

After setting up the MySQL database, the next step was to load in the data so that the classification model gets access for training and testing. Therefore, we first read in the data from a json file stored in a Jupyter Notebook called dev_mysql.ipynb. The data was then splitted into, what we called, train database and test database. Hereby, we reserved 1% of the whole data (114 samples) for the test database and the rest (11196 samples) for the train database. The train database is supposed to store the data so that a classification model then can use it for training and testing purposes in a supervised learning environment. The test database is used to simulate the user's input via the Writeback Extension.

Afterwards, the tables (train_db and test_db) in the database were created with the respective SQL queries:

```
train_db_query = "CREATE TABLE train_db (id int(6) NOT NULL AUTO_INCREMENT  
PRIMARY KEY, content longtext NOT NULL, target int(6) NOT NULL,  
target_name varchar(100) NOT NULL) "
```

```
test_db_query = "CREATE TABLE test_db (id int(6) NOT NULL  
AUTO_INCREMENT PRIMARY KEY, content longtext NOT NULL) "
```

The train database consists of an ID column that is additionally the primary key. Also, a content column which displays the e-mail content, along with a target column which shows the newsgroup label of the e-mail transformed into an integer from 0 to 19 and a column called target_name that displays the actual newsgroup label name.

For the test database, the table structure is built quite similar. The only difference is that it doesn't comprises the target and target_name column since it is supposed to test the Writeback Extension that is not given a label but rather the user should label the content by his own.

After creating the tables, a connection to the MySQL database was created and the respective train and test data sets were filled into the tables.

2.4. Data Preprocessing

Various preprocessing steps were performed to make the data in the train_db 'ready' for training the text classification model:

1. Lowercasing
2. Removing special characters, lines that contain 'https' and 'www.', line breaks and stopwords
3. Lemmatization

In addition to that, the data was transformed into a numerical representation using a tf-idf (term frequency- inverse document frequency) vectorizer. Including bigrams and trigrams into the corpus, each term is assigned a tf-idf value to measure how important it is in the document for a corpus of texts. The described steps for preprocessing the

data can be seen code-wise in the first part of train.py in which we apply the function 'preprocess ()' from text_preprocessing.py on the e-mail texts (content).

The whole dataset is then splitted into an 80% training set and 20% test set to respectively train the model and test its performance on an independent dataset.

3. The Algorithm

3.1. Supervised Text Classification

In order to demonstrate our use case using machine learning for text classification, we decided to choose a simple support vector machine classifier. The model's task is it to classify the e-mails and predict the newsgroup category. For the implementation of the support vector machine, we have used the scikit-learn library which can be seen in the second part in train.py after the preprocessing part. Since the focus was not on improving the model's performance in the first place, we used the default parameters of scikit-learn to train the model on the training set. But even without hyperparameter tuning, the results were relatively good for both the training accuracy as well as the test accuracy:

```
Train accuracy score: 99.82%
```

```
Test accuracy score: 88.90%
```

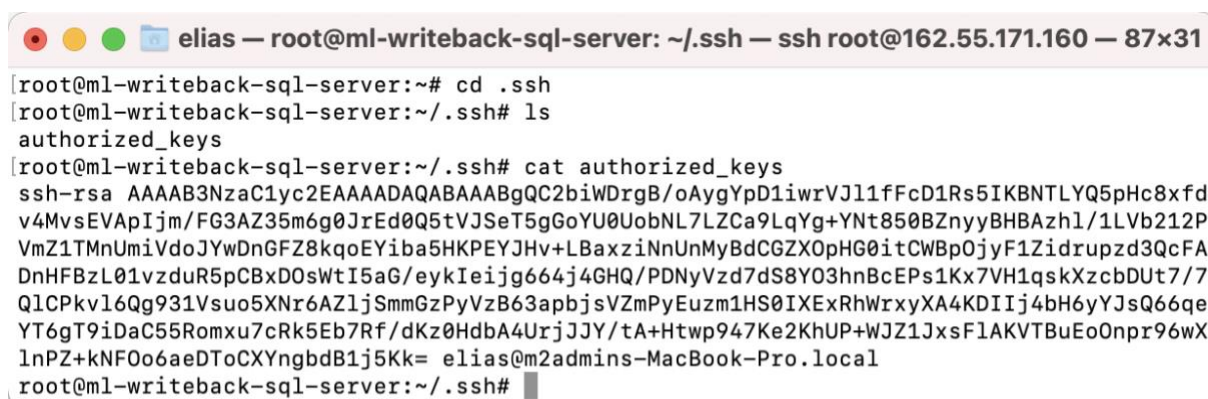
After training and testing on the respective datasets, the trained model and the tf-idf vectorizer, that was applied on the texts, were saved in a sav and pickle file for predicting the incoming e-mails via the Writeback Extension.

3.2. Server Deployment & Automation

After the Database was filled with data and the Python scripts were successfully tested locally, it was time to deploy the scripts to the server and to automate the execution in a way that new entries to the database, that simulate a new incoming email, will be classified automatically and the classifications with its corresponding probability will be written to the database. At this point, the split into a so called "test_db" and "train_db" is utilized. New incoming emails are simulated by transferring one or more rows from the test database to the training database. This leads to new rows in the train database that only have the "content" column filled. The columns target, target name and probability will be filled automatically by the Python scripts.

The files `classification.py`, `column_preprocessing.py`, `text_preprocessing.py`, `train.py` and the `requirements.txt` were copied from local storage to an own created directory on the server via the secure copy protocol (SCP). SCP uses SSH to authenticate the connection to the server.⁸ Since the public key of the developer's laptop was added to the authorized keys in the SSH directory of the server, as seen in the figure below, no password was needed for SSH authentication to login to the server and to copy files from local to server via SCP.

⁸ (IONOS, 2020)



```

elias — root@ml-writeback-sql-server: ~/.ssh — ssh root@162.55.171.160 — 87x31
root@ml-writeback-sql-server:~# cd .ssh
root@ml-writeback-sql-server:~/.ssh# ls
authorized_keys
root@ml-writeback-sql-server:~/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC2biWDrgB/oAygYpD1iwrVJl1fFcD1Rs5IKBNTLYQ5pHc8xfd
v4MvsEVApIjm/FG3AZ35m6g0JrEd0Q5tVJSeT5gGoYU0UobNL7LZCa9LqYg+YNt850BZnyyBHBAzh1/1LVb212P
VmZ1TMnUmiVdoJYwDnGFZ8kqoEYiba5HKPEYJHv+LBaxziNnUnMyBdCGZXOpHG0itCWBp0jyF1Zidrupzd3QcFA
DnHFBzL01vzduR5pCBxD0sWtI5aG/eykIeijg664j4GHQ/PDNYVzd7dS8Y03hnBcEPs1Kx7VH1qskXzcbDUt7/7
QLCPkv16Qg931Vsuo5XNr6AZ1jSmmGzPyVzB63apbjsVZmPyEuzm1HS0IXExRhWrxYXA4KDIj4bH6yYJsQ66qe
YT6gT9iDaC55Romxu7cRk5Eb7Rf/dKz0HdbA4UrjJJY/tA+Htwp947Ke2KhUP+WJZ1JxsFlAKVTBuEoOnpr96wX
lnPZ+kNF0o6aeDToCXyngbdB1j5Kk= elias@m2admins-MacBook-Pro.local
root@ml-writeback-sql-server:~/.ssh#

```

Figure 5: SSH authorized keys

Before executing the scripts, the requirements listed in the respective file needed to be installed with pip. Moreover, to ensure that text preprocessing as described in the foregoing chapter can be done, the nltk libraries stopwords, wordnet and punkt needed to be downloaded. This was done via the shell with the help of the nltk downloader.

In order to execute classification.py script, the train.py script needed to be executed once. The reason is that the classification script just loads a pretrained classification model as well as a pre-defined tf-idf vectorizer, which are trained and saved during the execution of the train.py script. As soon as this is done and all necessary files are saved, the classification.py script can be executed as much as needed. In order to regularly update the classification model as well as the tf-idf vectorizer and to constantly check for updates in the database that should be classified the ubuntu utility crontab is used. Inside the crontab file custom tasks can be scheduled. For our project the classification.py script is executed every ten seconds to check for new entries to classify and the train.py script is executed every Sunday at 4 AM to train the model on the current data in the database. As seen in the figure below, to schedule a task it needs

to be defined at what minute, hour, day of month, month and day of week a certain command should be executed. A star means that the command will be executed every minute/hour/day of month/month/day of week. Since crontabs parameters only allow to schedule a task every minute, 6 tasks were created to ensure the execution of the classification.py script every 10 seconds. Only the first of the 6 tasks will start immediately, because the other 5 tasks start with a sleep command that displaces the start time of the scripts to start one script every 10th second of the minute. The last line in the figure below states that the train.py script is executed at the 4th hour of the 7th day each week, which means every Sunday at 4 AM.

```

GNU nano 4.8 /tmp/crontab.ZDgfsX/crontab Modified
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * (cd /home/run/ml_writeback_db ; python3 classification.py)
* * * * * (sleep 10 ; cd /home/run/ml_writeback_db ; python3 classification.py)
* * * * * (sleep 20 ; cd /home/run/ml_writeback_db ; python3 classification.py)
* * * * * (sleep 30 ; cd /home/run/ml_writeback_db ; python3 classification.py)
* * * * * (sleep 40 ; cd /home/run/ml_writeback_db ; python3 classification.py)
* * * * * (sleep 50 ; cd /home/run/ml_writeback_db ; python3 classification.py)
* 4 * * 7 (cd /home/run/ml_writeback_db ; python3 train.py)

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line

```

Figure 6: Crontab

As can be seen in the classification.py script in the attached repository, the script first uses the Python MySQL connector to pull the data from the train database to check whether there are rows that do not have a value in the column “target”. This would indicate new rows, since already existing rows are classified and therefore have a value in the column “target”. Currently the whole database is checked for rows with the beforementioned criteria. Significantly more data in the database will lead to a loss of time performance when executing the script. Therefore, in a real environment it should

be estimated how much emails are potentially received within 10 seconds, because the script is executed every 10 seconds. The number of rows checked for new entries should be adjusted to that. However, currently the script takes 3.2 seconds to pull all rows, check them for new entries, to classify 2 new entries and to write the classification information to the database. If there are no new rows, the database connection that was opened to check for new rows is closed and the script ends running. In case there are new entries those new entries will be preprocessed and classified with the built model as described in the foregoing paragraphs. Afterwards, for each new email the predicted class number will be written to the "target" column, the corresponding name to the "target name" column and the probability of the classifications to the "probability" column. Again, before ending the script the database connection is closed. Now that all information is written to the database and classification is ensured, the classifications can be verified or changed by the user via the Writeback Extension.

4. *M2 Writeback Extension*

One of the core components of this project was integration of the writeback tool extension in the dashboard that could provide the user the opportunity to write insertions to the database directly from the Tableau dashboard interface. Tableau Writeback extension was developed by M2 and is offered as a part of M2 Tableau Toolbox solution.

4.1. *Functionality*

The extension has a number of specifications and features⁹. It is technically designed as Form that is to be integrated to Tableau dashboard. In the dashboard where the form is integrated, the user can interact with the form and insert updates to the database. The Form includes the number of elements that can be seen in Figure 7. The forms are created with the help of a simple editor which offers all the necessary elements. Those elements provide the functionality of layout design of the Form as well as the interface to set the values that are to be updated in the database. The Form is operated via a web interface on which all forms are created and managed. It is hosted on-site to prevent security issues that could arise from external connections.

⁹ (M2. technology & project consulting, 2021)

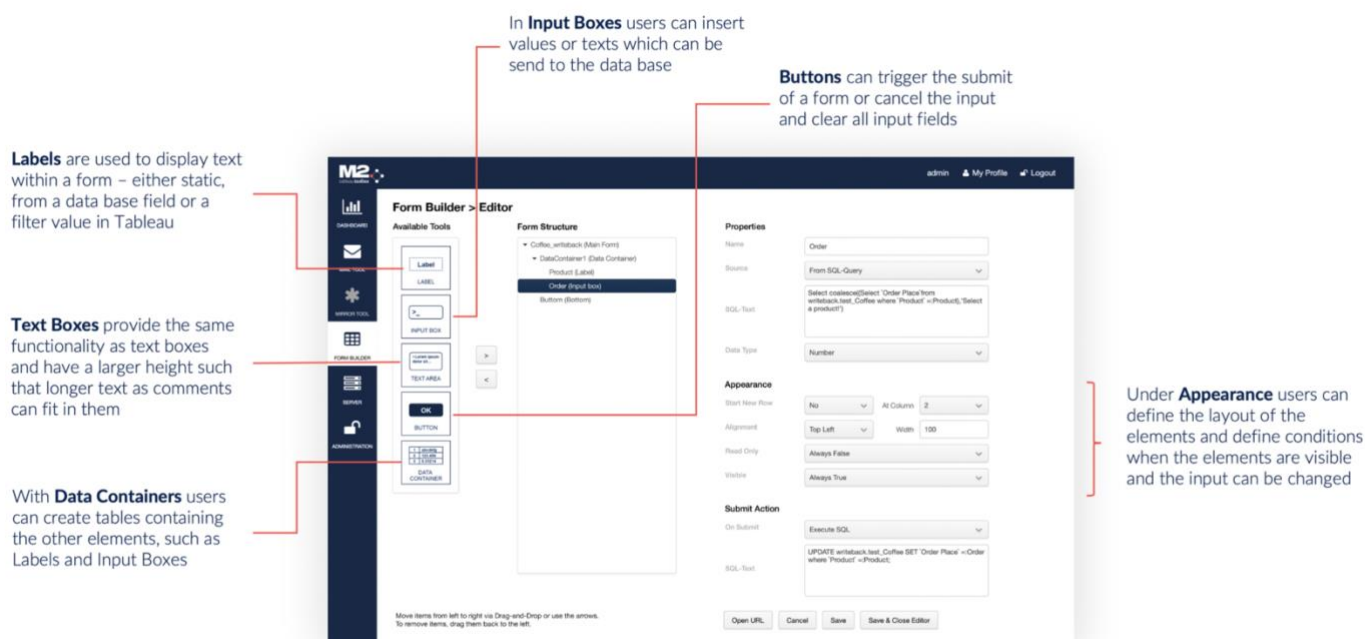


Figure 7: Writeback tool Form elements

With the above-mentioned specifications the writeback tool offers several functionalities¹⁰. As indicated previously the main functionality is **writing to the database** via Tableau Writeback Extension to reduce tool breaks and simplify processes. The form is designed in a simple way that does not assume any programming knowledge to configure it. This makes it particularly convenient for professionals working in the BI field without having extensive programming background. In terms of write-offs to the database, the tool only requires to provide

¹⁰ (M2. technology & project consulting, 2021)

the database connection details and write an SQL query that makes respective updates to the database.

All common databases (Postgres, MySQL, MSSQL, Oracle) are supported for writeback insertions.



Figure 8: Before and After – the possibilities of Tableau Writeback Extension

The writeback form offers the feature of **customization**. The design of the form can be customized to adapt to the dashboard needs via custom CSS. An important part of the writeback extension is its Security aspect. The Form Builder comes with its own individually configurable security concept. This ensures that write access to the data source is only possible in the designed context of the dashboard.

The Form Builder has **several versions** which are offered in the [Tableau Extension Gallery](#) or in the [AWS Marketplace](#) as part of the M2 Tableau Toolbox. M2 offers the Form Builder either in the cloud as a Software-as-a-Service solution or as part of local infrastructure. Additionally M2. technology & project consulting introduces new features to the extension and also offers customization of the form tailored to the specific use case.

4.2. Implementation

The overview of the integration of the writeback tool to the dashboard is depicted below.



Figure 9: Implementation overview of the Writeback tool

Creating a writeback form consisted of defining its layout, creating input parameters to interact with the dashboard as well as writing SQL queries to update the values in the target_name column. It was considered important for us to save the original category value (before writeback) to track changes, therefore the original value was written to the target_name_writeback column of the database table.

The following SQL query was integrated in the Writeback tool to update the values in the database.

```
UPDATE train_db
SET target_name_writeback = target_name, probability =1, target_name = CASE WHEN id = :InputBox_email_id
THEN :DropDown5 end where id = :InputBox_email_id;
```

Figure 10: SQL query integrated in the Writeback tool

The query updates target_name (email category) column with the value inserted by the user. The value can be picked from predefined drop-down selection to avoid irrelevant insertions to the database. Meanwhile, the original category value is saved into target_name_writeback column and the probability of category prediction is set to 1. It is assumed that the input of the user provides accurate category labels for the email, therefore the probability is set at 1.

The form enables the user to provide the email id for which the category label will be updated through the selection of value in drop-down list. Eventually, the layout of the writeback form is depicted below:

Please provide email id from the table on the left and update the category for the respective email. The original value in Category column will be saved in "Before Writeback" column.

Email ID:

Email category label:

✓

- Atheism
- Baseball
- Cars

Figure 11: Writeback form designed for 20NewsGroup email dataset

After completing the implementation steps, the technical output in the database looks like this (the column created to save the value before writeback is highlighted).

	ABC content	123 target	ABC target_name	123 probability	ABC target_name_writeback
1	From: ray@ole.cdac.com (Ray Berry)¶Subject	12	sci.electronics	1	comp.windows.x
2	From: alvin@spot.Colorado.EDU (Kenneth Alv	15	soc.religion.christian	1	rec.sport.hockey
3	From: urathi@net4.ICS.UCI.EDU (Unmesh Rat	5	comp.windows.x	1	rec.sport.hockey
4	Subject: NHL Summary parse results for gam	10	rec.sport.hockey	1	comp.windows.x
5	From: osburn@halcyon.com (Tim Osburn)¶Su	2	comp.os.ms-windows.misc	1	comp.os.ms-windows.misc
6	From: Steed.Bell@macrocosm.omahug.org (S	4	comp.sys.mac.hardware	1	[NULL]
7	From: aidler@sol.uvic.ca (E Alan Idler)¶Subje	15	soc.religion.christian	1	[NULL]
8	From: rap@coconut.cis.ufl.edu (Ryan Porter)*	1	comp.graphics	1	[NULL]
9	From: em@hprpcd.rose.hp.com (Electronic M	18	talk.politics.misc	1	[NULL]

Figure 12: Extract of the database output

4.3. Human-in-the-loop Machine Learning

Terms like Machine Learning and Artificial Intelligence are talked of across all industries, and it seems like everybody wants to enhance their business by using it. But only a few know what is behind these terms and how they work. Behind the “magic” are humans that trained the models by collecting and annotating terabytes of data. 90% of today’s machine learning applications are powered by supervised machine learning.¹¹ Supervised means, that the trained data was annotated.

Algorithms nowadays do not learn from programmers hardcoded rules anymore, they learn from examples and from feedback given by humans. Machine learning can automate and speed up the process of labelling data but needs to be backed up by humans when it is uncertain. So, the programmers task today is to create software that

¹¹ (Monarch, 2021)

allows to give feedback to the model by non-programmers. The overall goal of this is to increase the accuracy of the machine learning model.¹²

Taking this project as an example, the dashboard lists all classifications and the user can filter it for the certainty threshold to for example just list all classifications that are below a certainty of 80%. Certainty in this project is measured by the probability of the classification the model provides. Whenever the probability of a classification is below 80% the model asks for human feedback by listing the email in the dashboard. With the use of the writeback extension feedback is provided and written back to the database. But how does the model learn from that feedback to increase accuracy?

As described in chapter 3.2 the model is trained automatically once a week. This training is based on the data available in the database. Since the database also contains the newly classified emails, those emails also build the foundations for the model's training. Therefore, with more classified emails more data is trained to the model every week. Without the correction of wrongly classified emails those would be trained to the model, which then can lead to more wrong classifications, which will decrease the model accuracy. With the implementation of the Writeback Extension it is ensured that only correct classifications are trained to the model. Therefore, the model will increase in accuracy, because over time more data will be trained to the model. That means the model can base its decision on more example data provided during the training.

¹² (Monarch, 2021)

5. Outlook

Since this project was based on example data and had the purpose to highlight the functionality of the writeback tool, there are a few points to consider before implementing this solution into a real-world environment. First of all, it needs to be ensured that incoming emails or documents are automatically transferred into the database to add a new row for every incoming email or document. As already mentioned in chapter 3.2 it is advisable to estimate the incoming emails or documents within the timeframe the classification script is timed to be executed. The classification script then needs to be changed in a way that it only checks as many rows of the database for new entries as incoming emails or documents are expected. Moreover, the model performance can be monitored by saving the model's accuracy, number of samples that were trained and the time every time the model is trained. With this data, model performance can be visualized in relation to time and number of samples. Lastly, the Writeback Extension could also be used for unsupervised learning. Since the categories for the emails were given in this project, supervised learning was performed. In unsupervised learning no labels or annotations are given to the data that is subject to be classified. Already existing machine learning algorithms that try to find clusters for unannotated data could use the Writeback Extensions to receive feedback whether some datapoints are correctly clustered or belong to another cluster and learn from it. This process could be repeated until a reasonable way of clustering is found.

References

- Docker. (2021). *Docker Docs*. Retrieved from Install Docker Engine on Ubuntu:
<https://docs.docker.com/engine/install/ubuntu/>
- Docker Docs. (2021). *Install Docker Compose*. Retrieved from
<https://docs.docker.com/compose/install/>
- IONOS. (2020, May 15). *IONOS Digital Guide*. Retrieved from SCP (Secure Copy Protocol):
Was ist SCP?: <https://www.ionos.de/digitalguide/server/knowhow/scp-secure-copy/>
- M2. technology & porject consulting GmbH. (n.d.). *m2dot.com*. Retrieved from About M2:
<https://m2dot.com/en/m2/about-m2>
- M2. technology & project consulting. (2021). *M2 Tableau Writeback Extension*. Retrieved from
m2dot.com: <https://m2dot.com/en/tableau-writeback-extension#c2391>
- M2. technology & project consulting GmbH. (n.d.). *m2dot.com*. Retrieved from Tableau
Writeback Extension: <https://m2dot.com/tableau-writeback-extension>
- M2. technology & project consulting GmbH. (n.d.). *m2dot.com*. Retrieved from About M2:
<https://m2dot.com/en/m2/about-m2>
- Monarch, R. (. (2021). *Human-in-the-Loop Machine Learning. Active learning and annotation for human-centered AI*. New York: Manning Publications Co.
- MongoDB. (2021). *Comparing MongoDB vs MySQL*. Retrieved from
<https://www.mongodb.com/de-de/compare/mongodb-mysql>
- Selva86. (n.d.). *Github Datasets Newsgroup*. Retrieved from
<https://raw.githubusercontent.com/selva86/datasets/master/newsgroups.json>