



This project has received funding from the European Union's Horizon 2020 innovation action programme under grant agreement No 870373 – SnapEarth.

## Project 870373

H2020-SPACE-2018-2020

DT-SPACE-01-EO-2018-2020



## Deliverable D5.2

### Title: EarthSearch Service V1

<b>Dissemination Level:</b>	<i>PU*</i>
<b>Nature of the Deliverable:</b>	<i>OTHER**</i>
<b>Date:</b>	<b>26/10/2021</b>
<b>Distribution:</b>	<i>WP5</i>
<b>Editors:</b>	<b>QWANT</b>
<b>Reviewers:</b>	<b>CSR, isardSAT</b>
<b>Contributors:</b>	<b>Qwant, CERTH</b>

**Abstract:** This document presents the development of the SnapEarth EarthSearch service, and the additional components included. It documents the different ways to interact with it. Furthermore, this document provides the implementation of a question and answer database and the VQA method for communication with this database.

**\* Dissemination Level:** *PU= Public, RE= Restricted to a group specified by the Consortium, PP= Restricted to other program participants (including the Commission services), CO= Confidential, only for members of the Consortium (including the Commission services)*

**\*\* Nature of the Deliverable:** *R= Report, DEM= Demonstrator, Pilot, Prototype, DEC= Websites, patent filings, videos, etc., OTHER= Other, ETHICS= Ethics requirement, ORDP= Open Research Data Pilot, DATA= datasets, microdata, etc.*

## **Disclaimer**

---

This document contains material, which is copyright of certain SnapEarth consortium parties and may not be reproduced or copied without permission. The information contained in this document is the proprietary confidential information of certain SnapEarth consortium parties and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information in this document may require a license from the proprietor of that information.

Neither the SnapEarth consortium as a whole, nor any certain party of the SnapEarth consortium warrants that the information contained in this document is capable of use, or that use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using the information.

The contents of this document are the sole responsibility of the SnapEarth consortium and can in no way be taken to reflect the views of the European Commission.

## Revision History

Date	Rev.	Description	Partner
30/11/2020	1.0	Preparation of the document template	Qwant
	1.1	Introduction of CERTH's contribution	CERTH
30/09/2021	1.2	Updates with the comments from the review	Qwant
04/10/2021	1.3	Updates with the comments from the internal review	Qwant
20/10/2021	1.4	Template updates	Qwant, CERTH

## List of Authors

Partner	Author
Qwant	H. Randrianarivo
Qwant	H. Condemi
CERTH	M. Tsourma
CERTH	A. Zamichos
CERTH	E. Efthymiadis

## List of Reviewers

Partner	Author
isardSAT	M.J. Escorihuela
CSR	C. Cara

## Table des matières

Revision History.....	2
List of Authors .....	3
List of Reviewers .....	4
Table des matières.....	5
Liste of Figures .....	7
List of Tables .....	8
Glossary.....	9
1. Introduction.....	10
1.1 Purpose of the Document .....	10
1.2 EarthSearch Architecture overview. ....	10
2. EarthSearch services .....	12
2.1 Model service .....	12
2.2 Search service.....	13
3. Visual Question Answering System .....	17
3.1 Introduction .....	17
3.2 Data Sources.....	18
3.3 Scope & Functional Description.....	19
3.4 Architecture Description .....	20
3.4.1 EO Information Extraction Modules .....	21
3.5 Platform’s Tools Software and Hardware Specifications .....	27
3.6 Exposed Services Specification.....	27
3.7 Consumed Services and Usage Workflows.....	29
4. Recommendation and Accessibility services .....	30
4.1 Recommendation system.....	30
4.1.1 Introduction.....	30
4.1.2 Scope & Functional Description .....	32
4.1.3 Method Description.....	32
4.1.4 Platform’s Tools Software and Hardware Specifications .....	40
4.1.5 Exposed Services Specification .....	41
4.1.6 Consumed Services and Usage Workflows .....	43
4.2 Accessibility scoring system .....	44

4.2.1	Introduction.....	44
4.2.2	Scope & Functional Description .....	44
4.2.3	Method Description .....	46
4.2.4	Platform’s Tools Software and Hardware Specifications .....	49
4.2.5	Exposed Services Specification .....	50
4.2.6	Consumed Services and Usage Workflows .....	52
5.	Conclusion .....	53
	Bibliography .....	54

## Liste of Figures

Figure 1: EarthSearch conceptual architecture.....	11
Figure 2 Search definition defined for Vespa for the EarthSearch service .....	15
Figure 3: An example of a remote sensing image and relevant questions as implemented in the RSVQA approach proposed in [42] .....	20
Figure 4: Visual Question Answering System Architecture .....	21
Figure 5: Building footprint extraction results, “building” and “not building classes”, using an input image in RGB.....	24
Figure 6: VQA System UML workflow diagram.....	29
Figure 7: The procedure for queries recommendations based on their semantic similarity.....	33
Figure 8: xDeepFM architecture .....	38
Figure 9: (a) Extended xDeepFM with an NLP part, (b) Architecture of the NLP model .....	39
Figure 10: Example of produced training samples for xDeepFM recommender based on a sample query	40
Figure 11: Procedure for providing queries’ recommendations using the extended xDeepFM module ....	40
Figure 12: Recommendation system UML diagram – get similar queries .....	43
Figure 13: Accessibility scoring system’s architecture .....	46
Figure 14: Accessibility scoring system’s UML diagram. ....	52

## List of Tables

Table 1 Progress summary of the model service.....	12
Table 2 Progress summary of the search service .....	16
Table 3: Fire Detection Module Questions .....	22
Table 4: Flood Detection Module Questions .....	23
Table 5: Building Footprint Extraction and Damage Detection Module Questions.....	25
Table 6: Vegetation Status Detection Module Questions .....	26
Table 7: EarthSignature (CLC) Class Detection Module Questions .....	27
Table 8: VQA system’s specifications .....	27
Table 9: Description of the VQA’s Exposed Services: .....	27
Table 10: Recommendation system’s tools specifications .....	41
Table 11: Description of the Exposed Services.....	41
Table 12: Accessibility scoring system’s specifications.....	49
Table 13: Description of the Exposed Services .....	50

## Glossary

EO	Earth Observation
NLP	Natural Language Processing
CSF	Contrast Sensitivity Function
gRPC	gRPC Remote Procedure Calls
VQA	Visual Question Answering
API	Application Programming Interface
REST	REpresentational State Transfer
GPU	Graphics processing unit
CPU	Central Processing Unit
RPC	Remote Procedure Calls

# 1. Introduction

## 1.1 Purpose of the Document

This document aims to provide an overview of the advances in the implementation of the EarthSearch service.

The document is divided into five chapters. Chapter 1 briefly explains the document's structure and a reminder of the EarthSearch Architecture from deliverable 5.1. Chapter 2 describes the features we implement in the service and their status at the project's first review in March 2021. Chapter 3 describes the VQA system, which methods can be used to implement such a system, the architecture, and the services' interface. Chapter 4 covers the description of the Recommendation and Accessibility API. It describes the methods used for these services and their specifications. Finally, Chapter 5 concludes this document.

## 1.2 EarthSearch Architecture overview.

The Architecture schema Figure 1 describes the modules of the EarthSearch service. This service will be directly connected to the Qwant search engine. When a query is submitted on Qwant.com, it is first processed on the Qwant side then sent to EarthSearch if the query is related to the earth observation domain. We developed an intent detection model that recognizes EO-related queries for this purpose.

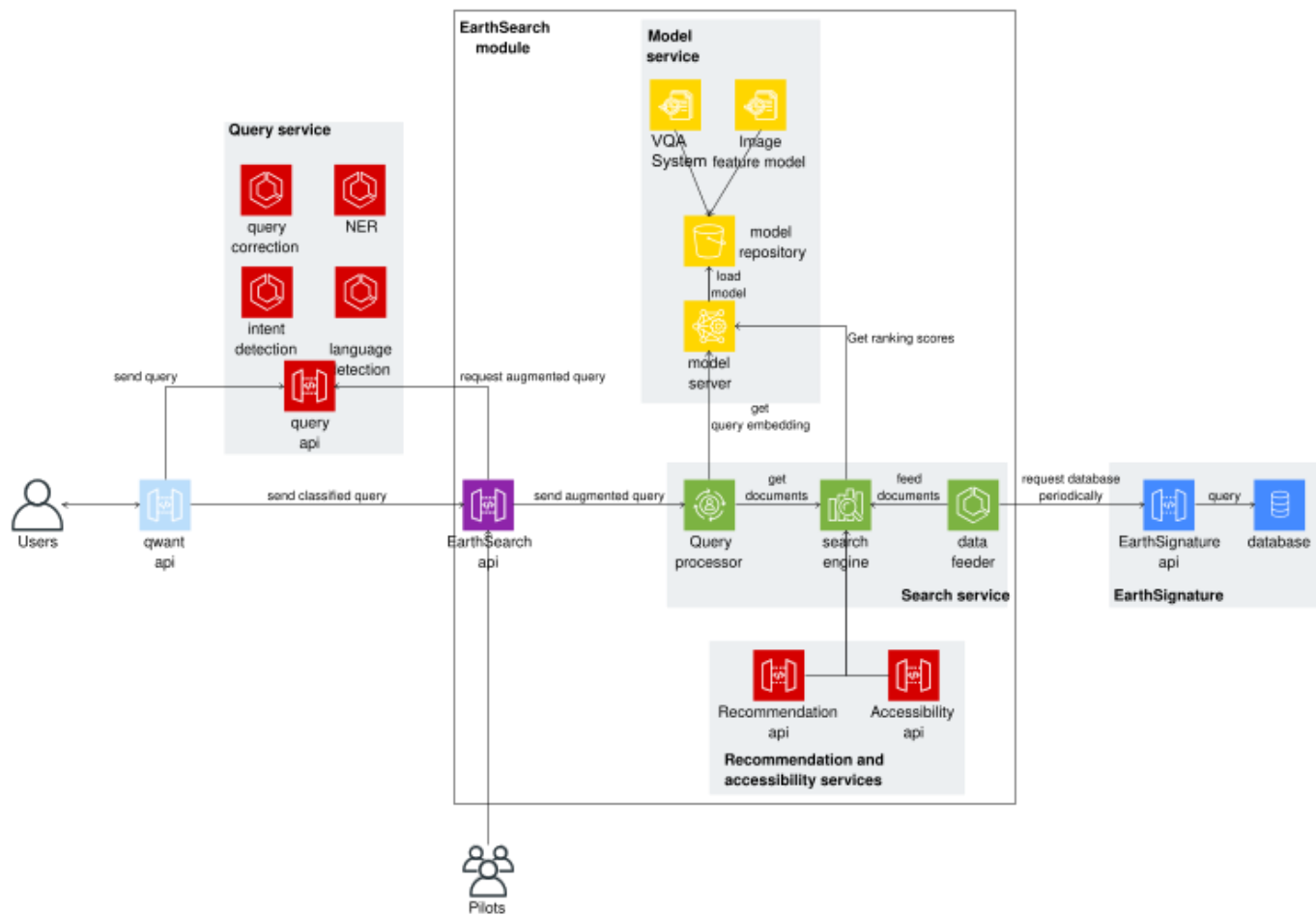


Figure 1: EarthSearch conceptual architecture

In our service, the query is sent to the search engine to retrieve the products processed by EarthSignature. When the first set of results is retrieved, we use a multimodal model to rank the products using the product's similarity with the query. Finally, the results will be displayed in an Instant Answers in the Qwant Results Page. To improve the ranking of its results, Earthsearch will use two API developed by CERTH in the v2 of the service. A detailed description of the VQA system and the Recommendation API are provided in Chapters 3 and 4.

## 2. EarthSearch services

### 2.1 Model service

To provide a high-quality service, EarthSearch makes heavy use of several machine learning models. The first is the Intent detection and classification model. This model is integrated directly into the Qwant Query API and aims at filtering queries related to the EO domain. It is this model that will redirect queries from the user to the EarthSearch Service. The second is a model that computes the similarity between an input query and the Search Service's data. The similarity is then used for re-ranking the results. For the moment, the preliminary ranking performances were not sufficient to validate a training method. For the next period, this topic will focus most of our effort.

Once the model is available, it can be challenging to serve it efficiently and update it without impacting the rest of the system. Most Machine Learning practitioners embed their model in an HTTP server and query it using a REST API. Following the best practices for serving a model at a large scale, in the SnapEarth project, we choose to use a dedicated server developed by NVIDIA Triton Inference Server. Triton Inference Server simplifies the deployment of AI models at scale in production. It is an open-source inference-serving software that lets teams deploy trained AI models from any local or remote framework on any GPU- or CPU-based infrastructure.

NVIDIA Triton server comes with a gRPC client library we can use to submit inference requests. However, the functionality of this client is low-level and not very user-friendly. We develop a wrapper around this client to query the server more efficiently for the project's specific needs. The following table summarizes our progress.

**Table 1 Progress summary of the model service**

Module	Status
Intent detection model	In Progress
Image - Text model	In Progress
Inference server	Finished: <ul style="list-style-type: none"> <li>- Server deployment</li> <li>- Models configuration</li> </ul>
Inference gRPC client	Finished
Model repository	Finished: <ul style="list-style-type: none"> <li>- Model repository for specifications</li> <li>- Repository deployment</li> </ul>

## 2.2 Search service

The Search service is the central component of EarthSearch. It is in charge of retrieving relevant results from its index according to an input query. To search for EO products accessible to everyone, we want to handle natural language queries, not only queries expressed by professionals working daily on EO Products.

To keep this promise, we divide the Search service into four components:

- A Data feeder
- A Search Engine
- A Query processor
- A gRPC API

The Data feeder is the component in charge of updating the SearchEngine index with new data. On the one hand, it queries the EarthSignature module periodically to retrieve new products. On the other hand, it sends the new set of data extracted from EarthSignature to the Search Engine to index them. EarthSignature also processes data by batch every day, so it is straightforward to retrieve new data.

The Search Engine organizes and allows efficient retrieval of the data indexed. It stores the data from EarthSignature as a document defined according to a predefined schema. The following figure describe a part of the schema we use:

```
document product {
```

```
  field productId type string {
```

```
    indexing: summary | index
```

```
  }
```

```
  field geometry type array<position> {
```

```
    indexing: attribute
```

```
  }
```

```
field productType type string {
```

```
  indexing: attribute
```

```
}
```

```
field cloudCover type float {
```

```
  indexing: attribute
```

```
}
```

```
field publicationDate type long {
```

```
  indexing: summary | attribute
```

```
}
```

```
field creationDate type long {
```

```
  indexing: summary | attribute
```

```
}
```

```
field quicklook type string {
```

```
  indexing: summary | attribute
```

```
}
```

```
field browseUrl type string {
```

```
  indexing: summary | attribute
```

```
}
```

```
field downloadUrl type string {
```

```
  indexing: summary | attribute
```

```
}
```

```

    field downloadUrl type string {
      indexing: summary | attribute
    }

    field segmentationArray type reference<segmentation> {
      indexing: attribute
    }

    field cloudMask type reference<segmentation> {
      indexing: attribute
    }
  }

```

**Figure 2 Search definition defined for Vespa for the EarthSearch service**

When the Search engine gathers the results, it ranks them using several signals extracted within the document, like the document's title or the product's creation date. Also, the search engine will use the scores returned by the Recommendation and Accessibility service to improve the relevance of the results for the users.

The Query processor is the service that handles the input query and extracts information to make the retrieval more efficient. Often queries need to be reformulated to improve the relevance of the documents retrieved. For this project, we will extend the general API for query expansion from Qwant with EO priors. It modifies the way the API finds synonyms and extracts named entities. With the new information added by the query expansion, the query's service format is understandable by the search engine.

The gRPC API will expose the search capabilities of EarthSearch to the pilots and Qwant.com. gRPC is a modern open-source, high-performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking, and authentication. To access the API, the API consumers must implement their client. The client's specifications will be defined using the protobuf format. The message exchange format is the same as EarthSignature, and the client will be able to query the API at earthsignature.snapearth.eu:443. The following table shows the status of the current implementation.

Table 2 Progress summary of the search service

Module	Status
Search gPRC server API	In Progress: <ul style="list-style-type: none"><li>- API definition not implemented</li><li>- Server endpoint not implemented</li><li>- Client not implemented</li></ul>
Query processor	In progress: <ul style="list-style-type: none"><li>- EO domain query expansion not implemented</li><li>- Query formatting not implemented</li></ul>
Search engine	Finished: <ul style="list-style-type: none"><li>- Vespa deployment: finished</li><li>- Document re-ranking finished</li></ul>
Data feeder	In Progress: <ul style="list-style-type: none"><li>- EarthSignature client implemented</li><li>- Data Formatter for Vespa document not implemented</li><li>- Vespa client feeder implemented</li></ul>

## 3. Visual Question Answering System

### 3.1 Introduction

Following the SnapEarth Grant Agreement description of work for the WP5, and more specifically objective T5.2 to develop a Visual Question Answering (VQA) method that “will allow anyone who wants to extract information from satellite images”, we propose the implementation of a VQA method that allows answering questions about the content of a satellite image. In the same scope, we define a set of questions applicable to satellite images and feasible to be answered adequately enough for use by both experts and non-experts.

Visual Question Answering (VQA) is a relatively new computer vision task where a system is given a text-based question about an image, and it needs to infer the answer. Typically, questions overlap with other tasks in computer vision with respect to semantic information, i.e. questions refer to sub-problems such as object recognition (e.g. What is in the image?), object detection (e.g. Are there any cats in the image?), attribute classification (e.g. What colour is the cat?), scene classification (e.g. Is it sunny?) and counting (e.g. How many cats are in the image?). Other more complex questions can be about the relationship among objects (e.g. What is between the cat and the sofa?) and common sense reasoning questions (e.g. Why is the girl crying?). It thus evident that VQA requires a system to do much more than other task specific computer vision algorithms, such as object recognition and object detection. As VQA encompasses many computer vision problems, a robust VQA system can even be considered as a component of a Turing Test for image understanding. Yet to construct such a robust system, we need algorithms that can rival humans in accuracy in all the involved computer vision tasks. Constituting such a challenge, the VQA domain has amassed a large amount of interest from the deep learning, computer vision, and natural language processing communities [31] [38].

Potential applications for VQA include assistive technologies for the blind and visually impaired individuals, enhancement of the human-computer interaction as a natural way to query visual content, and use as a system for image retrieval without using image meta-data or tags (e.g. to find all images taken in a rainy setting, we can simply ask “Is it raining?” to all images in the dataset). Apart from applications, VQA is an important basic research problem. A comprehensive VQA approach would permit the extraction of question-relevant semantic information from the images on multiple different levels ranging from the detection of miniscule details to the inference of abstract scene attributes for the whole image, based on the question [38], [42].

In the implementation of a Visual Question Answering System presented in this document, we aim to leverage the Earth Observation (EO) data - such as those that are accessible via remote sensing through the Copernicus Open Access Hub - in accordance with the objectives defined for the EarthPress, EarthAgriculture components of the SnapEarth platform. It must be emphasized that a VQA system for Earth Observation images - images obtained through remote sensing - differs significantly from the VQA systems for images that can be obtained in everyday settings and depict everyday objects and scenes.

Typically, a VQA model is made of four distinct components: 1) a visual feature extractor, 2) a language feature extractor, 3) a fusion step between the two modalities, and 4) a prediction component [38], [42]. In

our implementation, we focus on the visual feature extraction functionality (1) and the answers (predictions) it generates (4), aiming to propose a modular approach to extract information effectively belonging to different heterogeneous fields. The main reason for this design choice is the specialization level of the questions that are relevant to the domain of satellite image processing. Besides Natural Language Processing capabilities that would be employed in the language feature extractor component (2), have been implemented already by the other modules and services developed in the scope of the project. To compensate for the language feature extractor component functionality (2), we have created a pre-defined set of relevant questions that will be used to form the queries referring to the images. The set of questions will potentially be extended, in accordance with the development of the specialized modules of the platform.

### 3.2 Data Sources

In this implementation of a Visual Question Answering System will utilize the Earth Observation (EO) data that we will access primarily via remote sensing through the Copernicus Open Access Hub - in accordance with the objectives defined for the EarthPress, EarthAgriculture components of the SnapEarth platform.

Remote sensing is the acquisition of data from a distance. It involves the detection and monitoring of the physical characteristics of an area by measuring its reflected and emitted radiation via remote sensors, typically on satellites or aircrafts. Some examples of remote sensors are cameras on satellites and airplanes, sonar systems on ships, and cameras on UAVs. These remote sensors provide a global perspective and a wealth of data about Earth systems. Specific uses of remotely sensed images of the Earth include monitoring of large forest fires, monitoring of erupting volcanoes or dust storms, mapping of the rugged topography of the ocean floor, tracking clouds for weather prediction, the growth of a city or changes in farmland or forests over time. Remote sensing data constitute a valuable resource as they can support data-informed decision making in a plethora of domains concerning the current and future state of our planet [49] [50].

In the scope of the Visual Question Answering system, we utilize data from the dedicated satellites serving the Copernicus Programme that compound the Copernicus Sentinel families. Copernicus is an EU programme targeted at developing European information services based on satellite Earth Observation and in situ (non space) data, implemented by the European Commission (EC) with the support from the European Space Agency (ESA) for the Space component and the European Environment Agency (EEA) for the in-situ component. The Copernicus Open Access Hub provides complete, free and open access to Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5P user products, starting from the In-Orbit Commissioning Review (IOCR) [48]. More specifically, in the VQA system, we employ Sentinel-2 products as a source of EO imagery that will be input in the system.

The Copernicus Sentinel-2 mission comprises a constellation of two polar-orbiting satellites placed in the same sun-synchronous orbit, phased at 180° to each other. It aims at monitoring variability in land surface conditions, and its wide swath width (290 km) and high revisit time will support monitoring of Earth's surface changes. The Sentinel-2 Multispectral Instrument (MSI) samples 13 spectral bands: four bands at 10 meters, six bands at 20 meters, and three bands at 60 meters spatial resolution. The acquired data, mission coverage, and high revisit frequency provide for the generation of geoinformation at local, regional, national, and international scales. The data is designed to be modified and adapted by users interested in thematic areas such as spatial planning, agro-environmental, water, forest and vegetation, land carbon,

natural resource monitoring, and global crop monitoring. The Sentinel-2 data offer for the Open Access Hub consists of Level-1C and Level-2A user products [45] [46] [47].

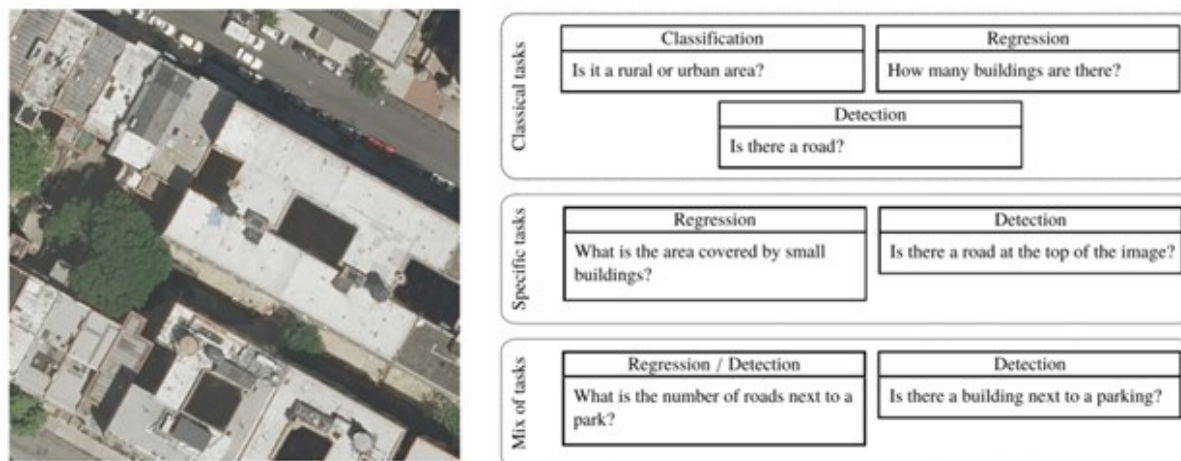
### 3.3 Scope & Functional Description

Although a lot of valuable information can be obtained from remote sensing images, specialized technical knowledge is also required to take advantage of the data, given a specific task. Moreover, depending on the type of the specific task, different types of approaches may be more or less suitable. Indicatively, in the context of the EarthPress component, automatic thresholding techniques are proposed for flood and fire detection, while a neural network approach is chosen for the building footprint extraction and damage detection task [39], [41]. A number of obstacles are present in the deployment of a unified generic approach for the exploitation of remote sensing images, including issues regarding the choice of methodology to apply and issues regarding the required input (type of sentinel product, image scale, number of images, etc.).

Remote Sensing Visual Question Answering (RSVQA) is an even newer task, recently introduced to enable generic and easy access to the information contained in remote sensing data by extracting it via a free form and open-ended questions. As of this moment, there is still a very limited amount of works that explore the use of questions formulated in natural language as a generic and accessible way to extract information from remote sensing images. Most notably, works [41] and [42] introduce the task, a novel dataset, and a methodology for producing questions for training, whereas [37] examines the automatic design of neural architecture framework for Remote Sensing Image Scene Classification (RSISC), a related task. Figure 3 demonstrates an example of a remote sensing image and the corresponding questions that the approach proposed in [42] aims to answer.

Considering the peculiarity of the remote sensing images, the types of questions that are relevant and useful in the RSVQA case have to be defined. In [42] [42], which is the work that most closely resembles our goal, authors experiment with five main types of questions, count questions (e.g. How many buildings are there?), presence questions (e.g. Is there a road?), area questions (e.g. What is the area of questions covered by small buildings), comparison questions (e.g. Are there more water areas than commercial buildings?) and rural/urban classification questions (using a generic definition of rural and urban areas). Figure 3 demonstrates such an example of a remote sensing image and the corresponding questions that the approach proposed in [42] aims to answer.

However, even when using this relatively restricted set of generic questions that only starts to cover the range of possible questions of interest, the need to use some kind of modular approach to tackle the different types of questions emerges as accuracies per question type can differ significantly. For example, counting questions pose a challenge and the use of dedicated components for these questions has been proposed [42], [43].



**Figure 3: An example of a remote sensing image and relevant questions as implemented in the RSVQA approach proposed in [42]**

To summarize, to the best of our knowledge, no methodology has been deployed yet that would permit us to satisfactorily exploit the EO remote sensing images (accessible via the Copernicus Sentinel satellites) to serve objectives defined in the SnapEarth context. However, in the scope of the SnapEarth project, a number of modules exploiting the EO data and addressing successfully very critical questions/points have already been developed or are being developed currently. Considering the above, and since there is no single algorithm or methodology for the implementation of an NLP-based VQA system that could cover an equivalent scope of points, we propose a modular approach for a VQA system based primarily on these EO information extraction modules and the functionalities that they offer.

More specifically, we aim to use the remote sensing inputs and outputs of the methodologies implemented in the scope of EarthPress, EarthAgriculture and EarthSignature to construct a predefined set of possible questions, and answers that we plan to incorporate into the VQA system.

In accordance with the wider SnapEarth objectives, potential domains of use of this VQA System include the agriculture and agricultural insurance domain, food security, disaster risk management, humanitarian interventions, and improving livelihood opportunities domains, the health and insurance domains, city management, spatial management, infrastructure management, civil protection, buildings construction, protected areas and forestry domains and journalism.

### 3.4 Architecture Description

In the Figure 4, we present the Visual Question Answering System we propose for the RSVQA task, in the context of the SnapEarth platform, that provides the capability to retrieve an EO image according to user input, select a question from a predefined set of questions and subsequently receive the answer. The function of the VQA system can be described in terms of the following steps:

**EO Data Retrieval:** The retrieval of the EO image is performed by the EO Data Retrieval functionality. Sentinel-2 products are downloaded from the Copernicus Open Access Hub through the Sentinel API, in accordance with the user input consisting of the date(s), the coordinates, or alternatively the location name,

of the point of interest, and optionally the radius around the point of interest. Even though we utilize Sentinel-2 data, the methodology presented could be applied to any geo-localized remotely sensed images.

**Question Selection & Module Selection:** The predefined set of questions consists of questions relevant to the EO images in the scope of the SnapEarth project such as disaster monitoring and vegetation monitoring. After the Question Selection by the user, the Module Selection of the suitable (i.e., the module that is capable of answering) for this specific question module is performed. The module is selected among the already implemented in the context of SnapEarth modules that we will henceforth be referred to as EO Information Extraction Modules and are described below in 3.4.1 EO Information Extraction Modules.

**EO Information Extraction (& Answer Generation) module:** After the Module Selection, the retrieved EO data are passed along to the EO Information Extraction Module that was chosen. The module performs its processing and outputs the results. The results are formatted and presented in the form of answers to the user.

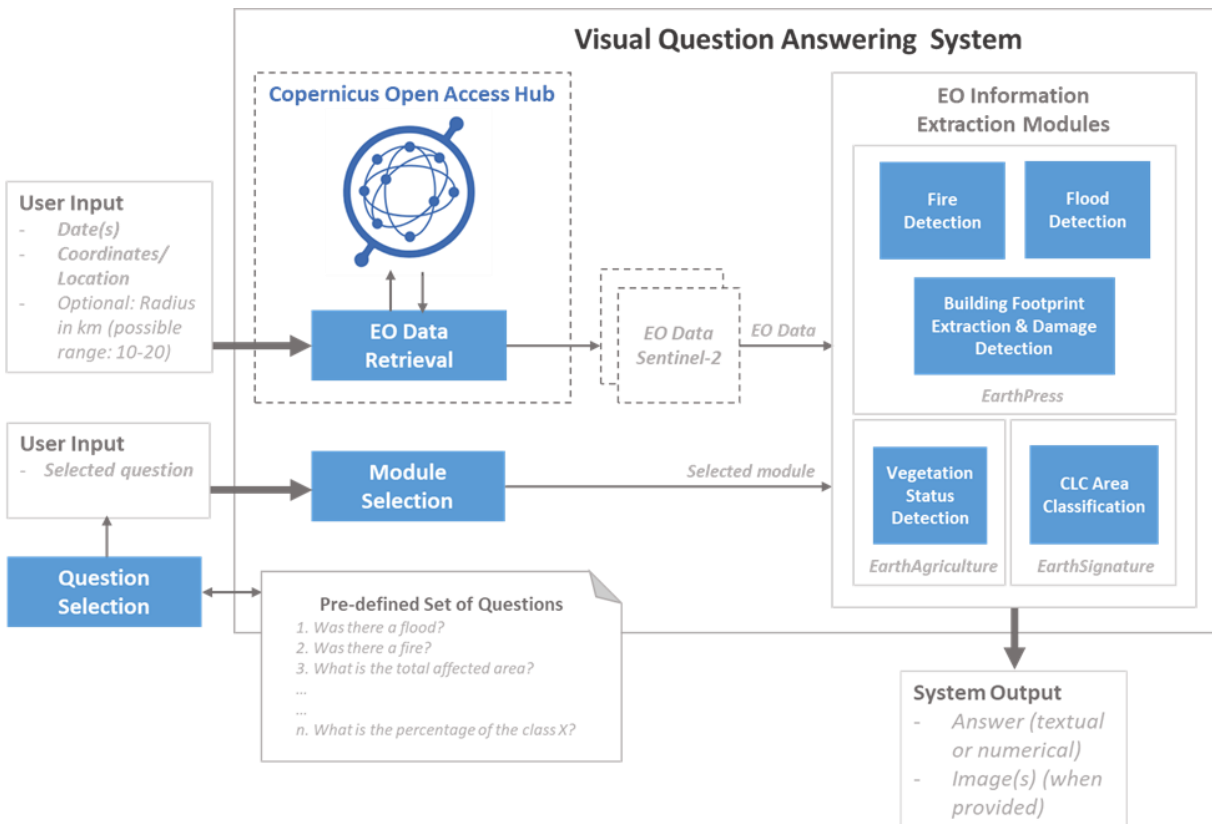


Figure 4: Visual Question Answering System Architecture

### 3.4.1 EO Information Extraction Modules

#### 3.4.1.1 Fire Detection Module

This module implements an EO data processing function, by providing the capability to process satellite imagery from time points preceding and following a fire event and creating a report about it consisting of

images and textual information for the post-fire scene understanding. The expected input for this module are two Sentinel-2 images, one before and one after the examined event, outputs include numerical and visual results regarding the areas affected by the fire. A more detailed description of the module's implementation is available in D6.1.

### 3.4.1.1.1 Questions

The questions that can be answered by the Fire Detection Module and will be available for selection by the end-user through the VQA system are listed in Table 3. The answer type *list of % percent values per class* is a list of pairs of land cover classes and the percentage corresponding to this class in the area of interest (e.g. [(Rice fields, 47,38%), (Inland marshes, 21.78%), (Salt marshes, 9.32%), (Estuaries: 2.40 %)]).

**Table 3: Fire Detection Module Questions**

ID	Question	Answer Type
1.	Was there a fire?	Yes/No
2.	What is the total affected area?	Number of km <sup>2</sup> & image
3.	What is the area classified as [high severity]?	Number of km <sup>2</sup> & image
4.	What is the area classified as [medium-high severity]?	Number of km <sup>2</sup> & image
5.	What is the area classified as [medium-low severity]?	Number of km <sup>2</sup> & image
6.	What is the area classified as [low severity]?	Number of km <sup>2</sup> & image
7.	What are the consequences in land cover?	A list of values, percentage (%) per class
8.	What is the uncertainty (error) of the calculations?	Percentage (%) for the input data, if possible, percentage % as per literature evidence

### 3.4.1.2 Flood Detection Module

This module implements an EO data processing function by providing the capability to process satellite imagery from time points preceding and following a flood event and creating a report about it consisting of images and textual information for the post flood scene understanding. The expected input for this module are two Sentinel-2 images, one before and one after the examined event, outputs include numerical and visual results regarding the areas affected by the flood. A more detailed description of the module's implementation is available in D6.1.

### 3.4.1.2.1 Questions

The questions that can be answered by the Flood Detection Module and will be available for selection by the end-user through the VQA system are listed in Table 4. The answer type *list of percentage (%) values, percentage (%) per class* is a list a of pairs of land cover classes and the percentage corresponding to this class in the area of interest (e.g. [(Rice fields, 47,38%), (Inland marshes, 21.78%), (Salt marshes, 9.32%), (Estuaries: 2.40 %)]).

**Table 4: Flood Detection Module Questions**

ID	Question	Answer Type
9.	Was there a flood?	Yes/No
10.	What are the consequences in land cover?	A list of percentage (%) values, percentage (%) per class
11.	Which land cover classes flooded?	A list of values, percentage (%) per class
12.	Which land cover classes has water retreated from?	A list of values, percentage (%) per class
13.	What is the uncertainty (error) of the calculations?	Percentage (%) for the input data, if possible, percentage % as per literature evidence

### 3.4.1.3 Building Footprint Extraction and Damage Detection Module

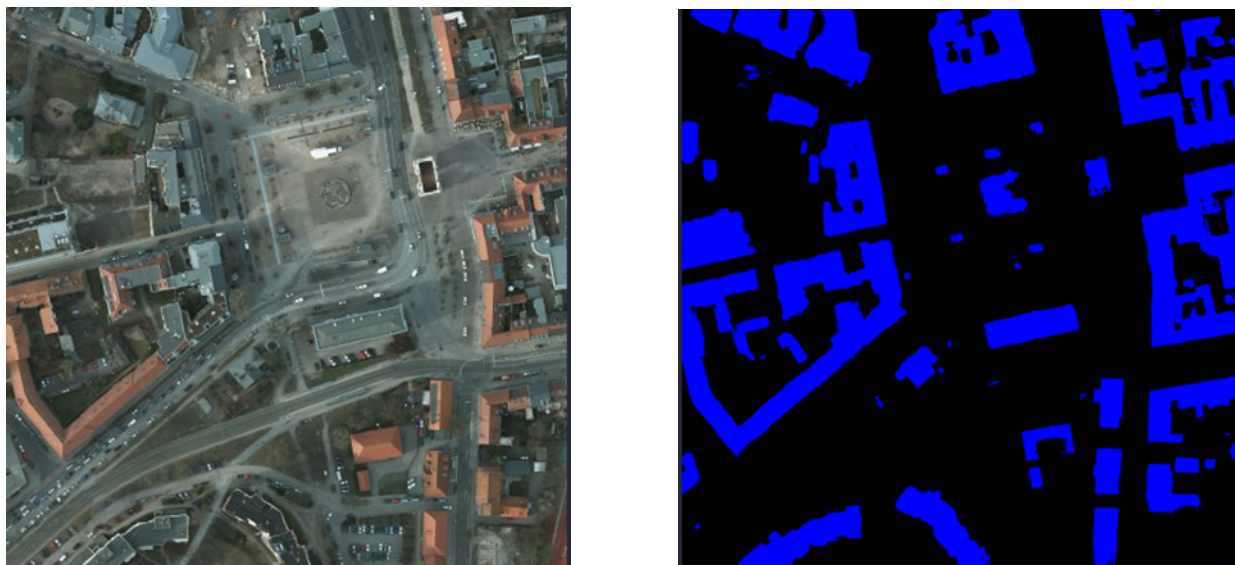
#### 3.4.1.3.1 Scope

This module implements an EO data processing function, by providing the capability to detect buildings in satellite imagery and the capability to detect changes in footprint of the building(s), given images from a time point preceding and a time point following a (disastrous) event (e.g. an earthquake). Images and textual information created will be used in the process of the final article creation.

#### 3.4.1.3.2 Functionality

For the current implementation of the building footprint extraction functionality, a natural RGB (satellite or UAV) image of the area of interest is sufficient. This image is fed into the SegNet deep convolutional neural network model for semantic pixel-wise segmentation [39], [40]. This approach allows the building(s) area segmentation on a pixel level, generating an output image with two (in our case) segmentation classes, “building” and “not building”, as in Figure 5 SegNet achieves 96.73% accuracy in our setup of this task. In

case no pixels (or no significant amount/percentage of pixels) are classified as “building”, we infer that no building is present in the image. For the damage detection functionality, data for two dates are required, according to logic one preceding and one following the disastrous event. The one preceding is used as a reference, whereas the one following is used to detect the changes that occurred. For both dates, the SegNet model is used to segment the image pixels into two classes, namely, “building” and “not building”. The results are then compared on a pixel level and the change in the area of the building is calculated.



**Figure 5: Building footprint extraction results, “building” and “not building classes”, using an input image in RGB**

#### 3.4.1.3.3 Data

Input data are RGB image(s). RGB imagery is sufficient for the functionalities provided by this module. Input images will be provided by the EO data access, and retrieval capabilities of EarthSearch will be passed on to the selected module. Output data produced after running the building detection functionality are a PNG image file of the segmentation map depicting the two classes, “building” and “not building”, and a text file presenting the results, the building footprint area, in square meters and as a percentage of the total area of interest, e.g., 14370.85 square meters and 15.97 %. All produced images (segmentation maps) and information will be passed on to the EarthBot component for the final article creation process.

#### 3.4.1.3.4 Related Work

The SegNet model, proposed in [33] and [34], is based on the architecture of VGG16 network. The novelty of SegNet lies in the manner in which the decoder upsamples its lower resolution input feature map(s). Specifically, the decoder uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. This eliminates the need for learning to upsample. The upsampled maps are sparse and are then convolved with trainable filters to produce dense feature maps. In literature, several studies have dealt with the issue of the Building damage detection either from simple RGB images or from satellite images. In [32], authors study how deep fully convolutional networks for semantic labeling can be adapted to deal with data beyond natural RGB images, namely, multi-modal and multi-scale remote sensing data. Their results indicate that late fusion of Digital Surface Model (DSM)

data with RGB data makes it possible to recover errors steaming from ambiguous data, while early fusion allows for better joint-feature learning but at the cost of higher sensitivity to missing data. They introduce the FuseNet architecture adaptable for both early and late fusion cases, obtaining results competitive with the state-of-the-art. In [40] the use of a U-Net-based semantic segmentation method for the extraction of building footprints from high-resolution multispectral satellite images is proposed, also achieving state-of-the-art results. The authors of [36] in their study focus on assessing damage levels for buildings, which is relevant to all types of natural disasters and can have a significant impact on the efficacy of search and rescue operations in their aftermath. In this study, the authors propose a new model called RescueNet, which is an end-to-end trainable, unified model to segment buildings and classify their damage levels. Another study concerning Building damage detection is the one presented in [51]. In this study, the authors present a deep learning-based computer vision model for detecting the magnitude of damage to buildings from pre- and post-disaster images. They use a convolutional neural network (CNN) in order to classify the dimensions of the damage and also evaluate other baseline models such as AlexNet. Apart from the studies implementing different methods for the Building damage detection task, in [35], the authors present a satellite imagery dataset for assessing building damage from satellite imagery. This dataset, called xBD, addresses the limitations levels existing by collecting data across eight disaster types, 15 countries, and thousands of square kilometers of imagery. Additionally, this study introduces a Joint Damage Scale that provides guidance and an assessment scale to label building damage in satellite imagery.

### 3.4.1.3.5 Questions

The questions that can be answered by the Building Footprint Extraction and Damage Detection module are listed in Table 5: Building Footprint Extraction and Damage Detection Module QuestionsTable 5 and will be available for selection by the end-user through the UI of the VQA system.

**Table 5: Building Footprint Extraction and Damage Detection Module Questions**

ID	Question	Answer Type
14.	Are there any buildings?	Yes/No
15.	What is the area of the building(s)?	m <sup>2</sup> and/or percentage (%)
16.	Is there any decrease in the building(s) area?	Yes/No
17.	What is the area of the damage?	m <sup>2</sup>
18.	What percentage of the building(s) area was damaged?	Percentage (%)

#### 3.4.1.4 Vegetation Status Detection Module

This module pertains to the EarthSignature component of the SnapEarth platform and implements a processing function of image data from EO by providing the capability to detect the vegetation status of the area of interest. The expected input for this module is a Sentinel-2 image of the area of interest. Outputs include numerical and visual results regarding the area. More information regarding the implementation of the vegetation status detection can be found in [56], [57], [58]. Note that in these documents, the term “processor” is used to describe what we refer to as “module”.

##### 3.4.1.4.1 Questions

The questions that can be answered by the Vegetation Status Detection module and will be available for selection by the end-user through the VQA system are listed in Table 6.

**Table 6: Vegetation Status Detection Module Questions**

ID	Question	Answer Type
19.	What is the vegetation status over the crop area?	Text, image samples of the vegetation status
20.	What is the percentage of the green vegetation?	Percentage (%) and image

#### 3.4.1.5 EarthSignature (CLC) Class Detection Module

This module pertains to the EarthSignature component of the SnapEarth platform and implements a processing function of image data from EO by providing the capability to classify the area of interest for any class in Corine Land Cover (CLC) nomenclature Level 2 and possibly Level 3. The expected input for this module is a Sentinel-2 image of the area of interest, while the output is the percentage of the contained CLC classes.

##### 3.4.1.5.5 Questions

The questions that can be answered by the EarthSignature CLC Class Detection Module and will be available for selection by the end-user through the VQA system are listed in Table 7.

**Table 7: EarthSignature (CLC) Class Detection Module Questions**

ID	Question	Answer Type
21.	Does the area of interest contain class X?	Yes/No
22.	What is the percentage of the class X (in the area of interest)?	Percentage (%)

### 3.5 Platform’s Tools Software and Hardware Specifications

The following table demonstrates the software specifications of VQA’s system.

**Table 8: VQA system’s specifications**

Licensing	A (non-open source) licensing of the development platform.
Core Implementation Technologies	Python.
Additional technologies utilised	pandas, numpy, SentinelAPI, shapely, rasterio, geojson, folium, PIL
Exposed APIs	RESTful web-services
Exchanged data format	JSON

### 3.6 Exposed Services Specification

**Table 9: Description of the VQA’s Exposed Services:**

Service Info	
Service’s Name	visualQuestionAnswering
Description	Web service, which given input parameters [geographical coordinates (latitude, longitude) or location name, date(s), radius (optional)] and an

	input question retrieves the image that corresponds to the input parameters and returns the answer to the question based on the image		
Method	POST		
Url	<protocol>://<IP>:<port>/ visualQuestionAnswering		
Headers			
	Name	Value	
	Accept	application/json, text/plain, */*	
	Accept-Encoding	gzip, deflate, br	
	Accept-Language	en-US,en;q=0.9	
Request Parameters			
	Name	type	Description
	Request's body	json	{ "coordinates": { "lat":37.983810, "lng":23.727539 }, "date": "2020-07-01T00:00:00Z", "radius": 10, "question ": "Are there any buildings?" }
Request Example			
	POST	http://localhost:4200/ visualQuestionAnswering	
Response			
	Status	Body	

	200	<p>Example of the response’s body :</p> <pre>{   "question": " Are there any buildings?",   "answer": "Yes" }</pre>
--	-----	---

### 3.7 Consumed Services and Usage Workflows

The VQA system will not consume any web service provided by the rest modules of the platform. It exposes the visualQuestionAnswering service. The results of the processing by the selected EO Information Extraction Module, numerical values, text, and images, will be returned and presented to the end-user. The described workflow is presented as a sequence diagram in Figure 6.

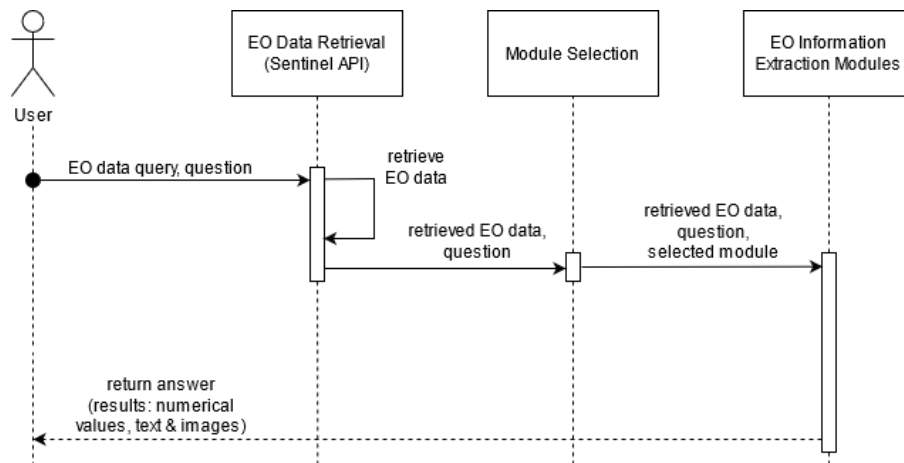


Figure 6: VQA System UML workflow diagram

## 4. Recommendation and Accessibility services

### 4.1 Recommendation system

#### 4.1.1 Introduction

Recommendation or recommender systems are systems that deal with the task of predicting the rating or the preference of a user over an item. Based on this, they recommend items to users that may be of their interest. The recommendation systems are commonly used in a variety of applications and platforms. They are used daily by web platforms/applications such as YouTube, Amazon, Netflix, and many more, for recommending content to their users. The kind of recommendations may vary from books, videos, movies to watch, products to buy, to similar search queries, articles that a user should read, etc. The type of recommended items can be adapted according to the services provided by each application.

One of their uses is as a filtering mechanism, which aims at filtering the huge amount of data that is available within or across a platform. By using a recommendation system with this scope and having as a basis the query that a user uses to search an item, the recommendation system will be able to filter the amount of available information existing for this query and provide to the user only the most related ones. However, the provided output from the filtering process sometimes may be irrelevant to the user's interest.

Moreover, a recommendation system might suggest items that the users might not have thought to search for on their own. In this way, recommenders are used in order to filter and present items that are within the user's preferences and/or to provide content that the user has not searched for explicitly but he/she will be interested in. Therefore, the recommenders are meant to facilitate users to find content according to their interests and preferences. Through this process, the recommendation systems achieve to improve users' experience and thus, increase user engagement in an application or a platform.

The techniques used by the recommendation systems can be grouped into three main types: a) content-based filtering, b) collaborative filtering, and c) hybrid methods. The content-based filtering methods, utilize a set of pre-defined characteristics or extract item properties and then calculate similarities between items. The recommended items are similar with previously liked or searched items by the user. For instance, if a user has watched one or more *Harry Potter* movies that are adventure, fantasy, and appropriate for families (based on IMDB) movies, a content-based filtering algorithm would recommend to the user other movies with similar characteristics, such as *Fantastic Beasts and Where to Find Them*.

Content-based methods take into consideration the features or behavior of a user given the item's features and the interactions/reactions a user has for these items (positive or negative reaction, or the ratings provided in a movie). These methods provide user independence through exclusive ratings that are used by the active user to build their own profile. In literature, several methods have been implemented and evaluated [5], including the Deep Knowledge-Aware network (DKN) [5]. The DKN is used mostly for click-through rate predictions<sup>1</sup> and in cases of news recommendation, it incorporates a knowledge graph

---

<sup>1</sup> Click-through rate prediction is a recommendation system's task concerning the prediction of the likelihood that something on a website (e.g. an advertisement) will be clicked.

representation. Additionally, one more widely used method is the LightGBM [6]. LightGBM is a popular method that makes use of tree-based learning algorithms and is used in literature in many machine-learning tasks, such as multi-class classification and click-through rate predictions.

The collaborative filtering methods are based on the collection and analysis of all the available information that concern one user's preferences, activities, and interactions and recommend items based on this user's similarity with other users [5]. In particular, these methods utilize both user's previous behavior (items previously liked/purchased/clicked/watched) and the behavior of other users of the same platform. In this way, the system can suggest items to a user A, that have been liked by other similar users even if the user A has not liked any item similar to this item.

In literature, collaborative-filtering methods are used in a great extend due to their nature and results [7]. Neural Collaborative Filtering (NCF) [8] model is a deep learning algorithm with enhanced performance for implicit feedback used in many cases, such as items recommendation and click through rate predictions. This model uses matrix factorization and leverages a multi-layer perceptron to learn the user-item interaction function. An additional method for collaborative filtering is LightGCN [9]. LightGCN is a Deep learning algorithm that simplifies the design of GCN for predicting implicit feedback. Methods for capturing both long and short-term user preferences have been implemented, such as GRU4Rec [10]. GRU4Rec is a sequential-based algorithm that aims to capture both long and short-term user preferences using recurrent neural networks. Each method follows a different approach with its own advantages and limitations. According to data availability and the scope of the system that will be implemented, the most convenient approach should be followed.

Finally, the hybrid methods are referred to methods that combine methods from both content-based and collaborative filtering, considering both the preferences of similar users as well as the properties of similar liked items. Hybrid methods could be more effective in some cases and are mostly used to aggregate collaborative filtering and content-based filtering to improve recommendation system's accuracy [5]. Wide and Deep [11] is a deep learning algorithm that can memorize feature interactions and generalize user features. Another method is LightFM [12], which is a hybrid matrix factorization algorithm for both implicit and explicit feedback.

The xDeepFM [13] is a deep learning-based algorithm that can jointly learns explicit and implicit<sup>2</sup> high-order feature interactions effectively and requires no manual feature engineering. The generation of feature interactions in an explicit fashion is caused by the use of a novel new layer introduced, called Compressed Interaction Network (CIN). The advantages of using this new layer lie in the use that high-order feature interactions are measured explicitly and that the complexity of the network does not grow exponentially with the degree of interactions. Another advantage of this network is that it can learn low- and high-order

---

<sup>2</sup> Implicit feedback is the data gathered from the user's behavior through the interactions s/he has in genera. Implicit feedback does not include ratings or specific actions, it includes numbers indicating the items a user purchased, the user's browsing history, search patterns, how many times s/he has played a song or watched a movie, how long s/he has spent reading a specific article etc.

feature interactions simultaneously from the input raw features. Due to these advantages, xDeepFM is a widely used model that can be further used and evaluated on different recommendation tasks.

#### 4.1.2 Scope & Functional Description

The scope of web search engines is to provide a collection of data to users upon request. The request is, in most cases, in the form of input text that is commonly referred to as a query. Web search engines try to retrieve from their databases the most related content with the input query and present it to users, while in parallel ranking the retrieved content based on its relevance with the query's topic. As an extension to the contemporary web search engines functionality, and in order to facilitate users' searches, recommendation systems can be used for this purpose.

The recommendation systems can suggest similar queries with the initial input query of the user. These similar queries are retrieved from historical queries that other users had searched in the past. By recommending similar queries, the system may suggest queries to the user that s/he might not have thought to search for or to provide enriched queries, similar to the initial ones, that may lead to better information retrieval. This process may improve users' experience on the platform and thus increase users engagement.

Following this and within the scope of addressing objective 7 “Build an EO data access portal for the general public based on natural language processing and strongly integrated with the Qwant search engine” of the SnapEarth project, a recommendation system for user queries have been developed within Task 5.2 which can act as an assistive module that will facilitate the EO data retrieval. The scope of the recommender is to provide similar queries with the user's input query, based on other searches that have been performed through the platform in the past, which will be integrated with the Qwant search engine. By using NLP and recommendation system techniques, the recommender will facilitate the EO data retrieval from the general public by recommending previously searched queries by other users or augmented queries targeting the retrieval of more relevant results. The communication of the recommendation system and the rest modules of the EarthSearch platform will be established through the exposed web services of the recommendation system, which will be described in detail in the next sections. The tuning and the evaluation of the proposed methods will be presented in detail in Deliverable 5.3.

#### 4.1.3 Method Description

For the recommendation of queries that are similar to the user's input query, two AI-based query recommendation mechanisms have been developed, based on NLP technologies on the content of the queries that can be integrated into QWANT's web search engine and are described in this section. In particular, the first one regards a mechanism that produces complete queries based on the past historical searches by analyzing the similarity & the correlation between them. In order to succeed in that, various sentence representation methods are utilized. The second method provides query augmentation, by extending the typed query with the most correlated tokens from past queries. The latter method uses and extends the eXtreme Deep Factorization Machine (xDeepFM) [13] recommendation algorithm.

#### 4.1.3.1 Sentence similarity-based recommender

The first method that was implemented as part of the queries’ recommender is based on the sentence similarity of the input query with the historical queries. With the term of sentence similarity or semantic textual similarity, we refer to the measure that indicates the degree of similarity between two different texts.

Figure 7 describes the whole procedure for recommending similar queries. The procedure consists of two (2) phases. At the first phase, historical queries that have been searched from the platform’s users are pre-processed and transformed to vectors of real numbers, also known as embeddings’ representation. The produced embeddings are then stored in the system’s database. The second phase refers to the retrieval of similar to the input text queries. In this phase, the pre-processing, the vectorization, and the comparison of the input text, with the already processed queries take place. A similarity score is calculated for the input sentence and each one of the historical queries. Based on the computed similarity scores, the N most similar queries<sup>3</sup> are the final recommendations and are sent to the user.

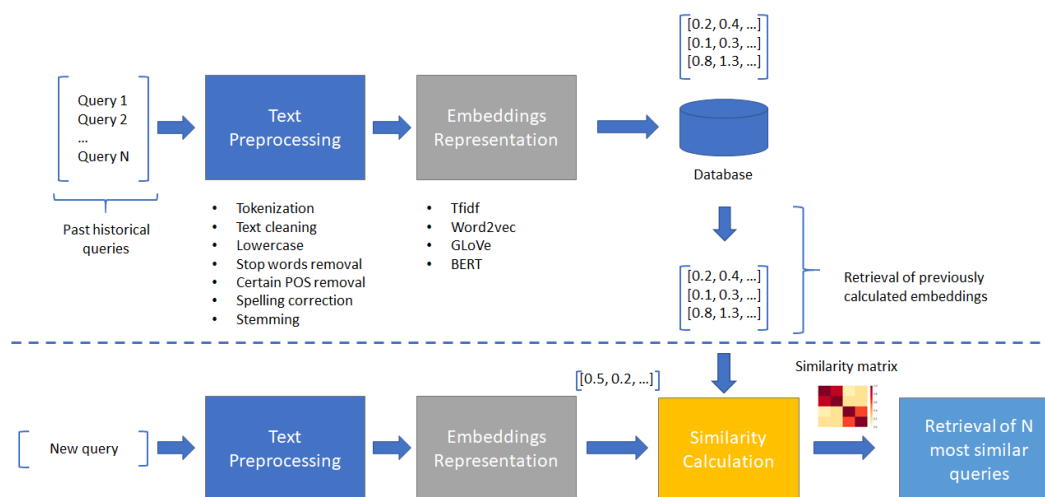


Figure 7: The procedure for queries recommendations based on their semantic similarity

The above procedure consists of the four (4) processing steps explained in detail below.

1. **Text pre-processing:** refers to the actions of text cleaning and filtering that precede the text’s vectorization. The text pre-processing step usually contains: the tokenization<sup>4</sup> of the input text, the transformation of all text’s characters to lower case, the removal of the stop words<sup>5</sup>, cleaning of

<sup>3</sup> Similar queries are the ones with the highest similarity scores. The value of N is configurable, and can be updated according to the requirements that each platform has.

<sup>4</sup> Tokenization is the process of splitting a text into several words/tokens.

<sup>5</sup> The stop words are the most commonly used words of the language such as in, the, at, who, he, she, etc.

text from punctuations and non-alphabetical tokens, the removal of certain part of speech (POS) of the language, the spelling correction and stemming<sup>6</sup> of the words included in the input query.

2. **Embedding's representation:** refers to the step of transforming/representing an input text into a vector of real numbers. The outcome vectors that represent words or phrases are usually referred to in the field of Natural Language Processing (NLP) as word or sentence embeddings. The intuition behind the embeddings is that we try to learn representations for words or sentences that are similar for words or sentences with the same meaning. There are several works in the literature that deal with the problem of representing words or sentences as embeddings. In the presented recommendation system, we have implemented four (4) different methods for producing sentence embeddings.

#### a) TF-IDF sentence embeddings

In information retrieval, the term frequency-inverse document frequency of (TF-IDF) refers to a statistical procedure that is commonly used for revealing how important is a certain token/word for the retrieval of a document.

The TF-IDF method calculates the value of each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears in. In this way, words with high TF-IDF values depict a strong relationship with the document they appear in. Considering this, if a word with high TF-IDF value appears in a query, the documents that contain this word could be of the user's interest [19].

Moreover, the TF-IDF method is commonly used as a weighting factor in different tasks of the field of NLP (e.g., text classification and semantic sentence similarity). Specifically, this is a numerical measure that expresses how relevant a word is to a document in a collection. It is a combination of two measures: TF that counts the occurrences or frequency ( $f$ ) of the term ( $t$ ) in a document ( $d$ ) and IDF, which measures how much information the term  $t$  provides across all documents ( $D$ ).

$$TF(t, d) = \frac{f_{t,d}}{\max \{f_{t',d} : t' \in d\}} \quad (1)$$

where,  $f_{t,d}$  is the count of the term  $t$  in a document  $d$

$$IDF(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (2)$$

---

<sup>6</sup> Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form.

where  $|D|$ , is the total number of documents in a corpus  $D$ . The final TF-IDF score is given by the multiplication of the  $TF$  and  $IDF$  terms.

$$TF - IDF(t, d, D) = TF(t, d) \cdot IDF(t, D) \quad (3)$$

Thus, the TF-IDF score of a term increases proportionally according to the number of this term's occurrence within a document, but it offsets based on the occurrence of this term on the rest documents of the corpus.

For the needs of the queries recommender and for producing the sentence embeddings of the queries when the TF-IDF method is used, the `TfidfVectorizer` class of the scikit-learn package [20] were utilized.

However, TF-IDF does not consider words similarities. Therefore, the embeddings' representation is mainly based on the existence of the words within the sentence without considering the existence of similar words. In order to tackle this problem, we also implemented a method that utilizes pre-trained word embeddings (Word2vec and GloVe), which are analyzed next.

#### **b) Word2vec-based sentence embeddings**

Word2vec is a method for learning word embeddings representation from a huge corpus of text. It was introduced by Mikolov et al. [21] and is a method that learns to represent words as a vector of real numbers. Words with the same or similar meaning tend to be represented by similar vectors. In the presented recommendation system, for the representation of words with embeddings, we used the pre-trained embeddings that are provided by Google in their archive [22] and are trained on part of Google News dataset. The model contains 300-dimensional vectors for 3 million words and phrases. In order to extract the final sentence embedding, we combined the word embeddings of the words included in a sentence with their TF-IDF score by calculating the weighted average of all the word embeddings using as weight each word's TF-IDF score.

#### **c) GloVe-based sentence embeddings**

A similar approach with the *Word2vec-based sentence embeddings* was implemented, but instead of using the Word2vec pre-trained embeddings, we utilized the GloVe embeddings. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus. GloVe was introduced by Pennington et al. [23]. Within the scope of the project, we used the provided by the Stanford NLP group pre-trained word embeddings that are available in [24]. In order to compute the final embeddings of the sentences, we used the same approach as in Word2vec-based sentence embeddings. Therefore, the final sentences' representations were weighted averages of the word embeddings of each token included in a sentence, while using as weights of each word their TF-IDF score.

#### d) BERT-based sentence embeddings

The fourth method that was implemented for the comparison of sentence queries utilizes the embeddings produced by Bidirectional Encoder Representations from Transformers (BERT) model. BERT was introduced by Devlin et al. [25] and achieved state-of-the-art performance in many NLP tasks, including question answering, sentence-pair completion, sentiment analysis, and others.

BERT has been built on top of Transformer [44], an attention mechanism that learns contextual relations between words or sub-words within a text. However, BERT is differentiated from the Transformer by using only the Encoder mechanism while the latter utilizes both an Encoder and a Decoder part.

The key difference of BERT is its way of training. BERT is pre-trained, as a language model, on a huge corpus of unlabeled text. For the pre-training corpus, the BooksCorpus (800M words) [28] and English Wikipedia (2,500M words) were used. While previous models that utilized Transformer performed their training by reading a text sequence either from left to right or combined left-to-right and right-to-left, BERT performed a bi-directional training of Transformer. In this way, the model was able to learn the context of a word within a sentence by looking at all its surrounding words (both from left and right of the word). For training BERT, two training tasks were utilized:

- the “masked language model”, which randomly masks some of the tokens from the input, and the objective is to predict the masked words based only on its context
- the “next sentence prediction” task that pre-trains the model by predicting if a sentence B is the following/next sentence of a sentence A. Through the above training procedure, BERT is able to grasp patterns in language, something that has been proved to be able to empower the model when it is used, after fine-tuning, in downstream tasks.

BERT can be also used for providing word and sentence embeddings. In contrast with the previously analyzed methods (i.e., TF-IDF, Word2vec, GloVe) that have a fixed representation for each word, BERT produces words representations that depend on the rest words of the sentence. Thus, a word can be represented by a different embedding based on the meaning of the sentence. For instance, the representations of the word “*park*” in the following sentences: “I am going to **park** the car near the bus station.” and “We are going to go to the **park** for a walk”, will differ when using BERT model, while using TF-IDF, Word2vec or GloVe models the results will be the same.

The above methods presented will be evaluated on existing datasets. The evaluation results will be presented in Deliverable 5.3.

3. **Similarity Calculation:** in order to compare the similarity of the embeddings of two sentences we use the cosine similarity [26], which is a similarity metric between vectors that is usually used in the field of NLP. The cosine similarity measures the cosine of the angle between two vectors, which

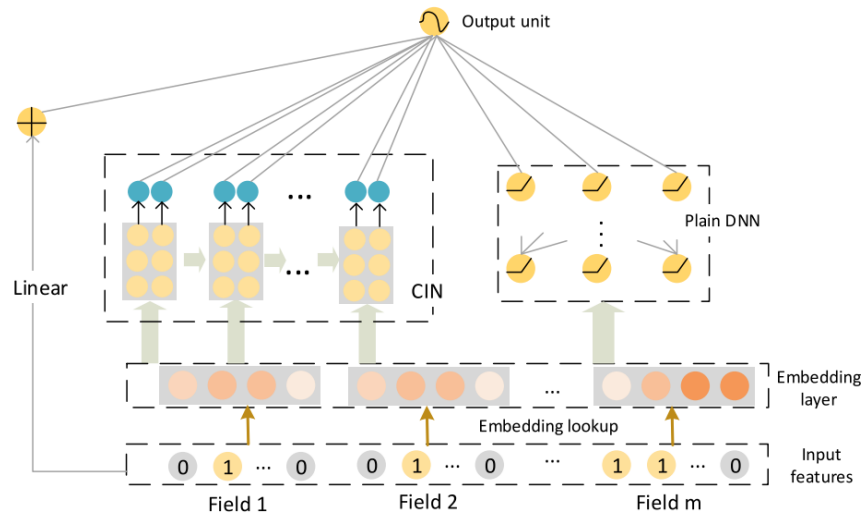
is also the same as the inner product of the same normalized vectors. The cosine similarity of two N-dimensional vectors  $\vec{v}$  and  $\vec{w}$  can be calculated as follows:

$$\text{Cosine similarity } (\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i \times w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}} \quad (4)$$

4. **Retrieval of N most similar queries:** The final step for recommending queries is the retrieval of the N most similar queries with the input query. After having vectorized the input text with one of the aforementioned methods, the recommender calculates the similarity of the embedding of the input query with the embeddings of other queries stored in the system’s database, by using the cosine similarity as described before. The recommender, then, finds the N most similar queries (queries whose cosine similarity score with the input query is higher) and returns their textual information to the user.

#### 4.1.3.2 xDeepFM-based recommender

The second method that was implemented for the purpose of the recommender system is based on the xDeepFM model. The xDeepFM is a hybrid model that combines characteristics from both content-based and collaborative filtering methods. The evaluation of the xDeepFM showed state-of-the-art results in different datasets [27]. The xDeepFM introduces a Compressed Interaction Network (CIN), which is combined in this model with a classical deep neural network (DNN) and a linear model. Figure 8 presents the xDeepFM architecture as it was presented in [27]. The linear model learns the patterns of the raw input features, the DNN, contributes by learning sophisticated and selective feature interactions implicitly, while the CIN part allows vector-wise feature interaction inference, rather than bitwise, in contrast with previously proposed methods in literature [53] []. The model can treat as input multi-field categorical data in the form of feature arrays. The output of the model is binary probability estimates. The xDeepFM has been used for recommendations on user-item datasets such as Movielens [52], in which both users’ information (e.g., userId, age, gender, occupation, etc.) and items information (e.g., movieId, genre, etc.) are available.



**Figure 8: xDeepFM architecture**

However, even that this approach is convenient for single-value and multi-value categorical features, it lacks on the way of treating textual data. Following this and in order to be able to leverage the user’s NLP data, we extend xDeepFM with an NLP part for modeling more efficiently textual data (Figure 9), aiming at recommending relative to the input queries’ terms. The NLP part takes as input textual data and the input sentence is passed through an NLP model, which is presented in Figure 9 and which provides as an outcome the representation of the input sentence as an embedding of  $d = 200$  dimension. The produced embedding is used as an extra input field to the xDeepFM model. The NLP model of Figure 9 consists of a bi-directional LSTM [30] with an attention layer [29].

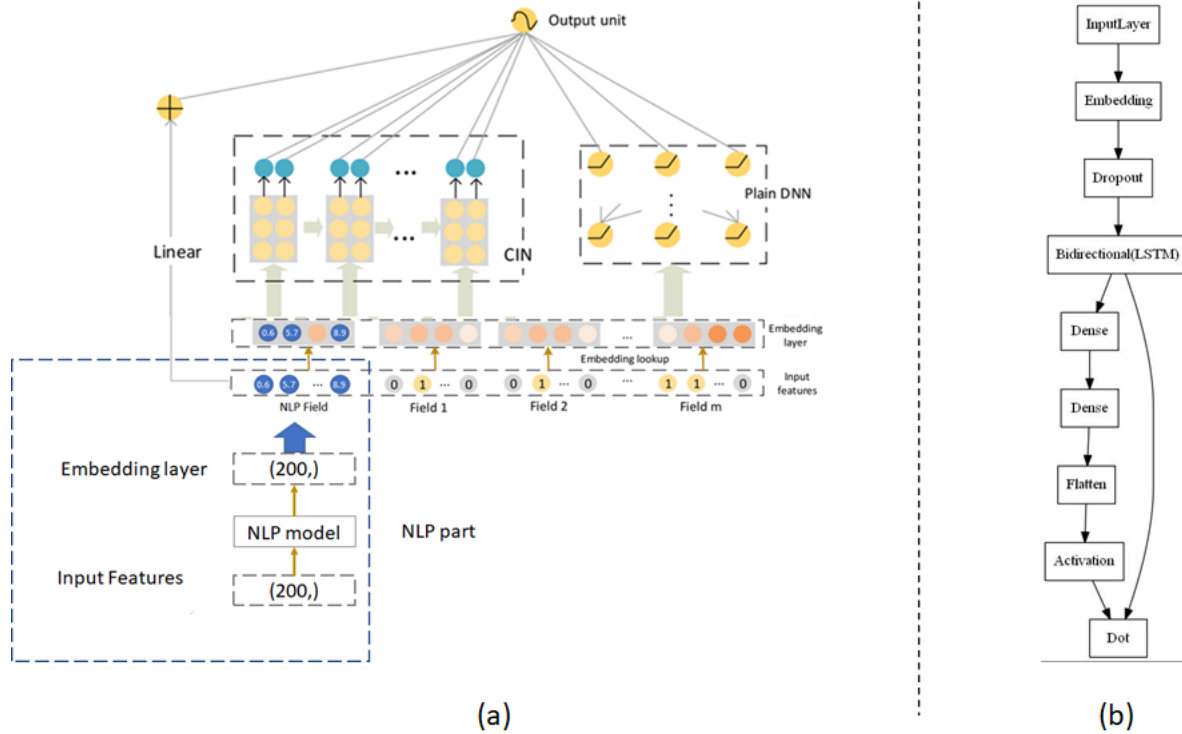
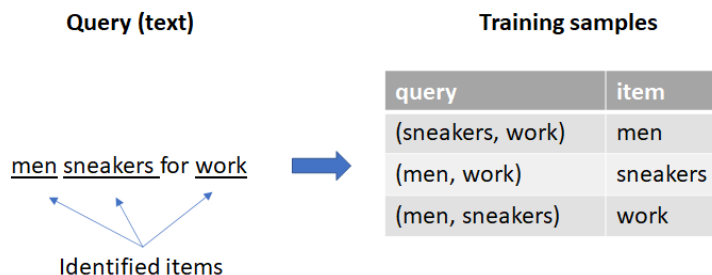


Figure 9: (a) Extended xDeepFM with an NLP part, (b) Architecture of the NLP model

The extended xDeepFM model (Figure 9) will be used in order to provide related queries to an input query. As items, we define the  $N = 10000$  most frequent nouns (a parameter that is easily configurable) that are within the training set of queries. In order to extract the most frequent nouns within the training set of queries, we apply part-of-speech (PoS) tagging in the whole training set and keep only the tokens that are labeled as nouns. The frequency of each token is also calculated and the final set of items consists of the  $N$  most frequent items.

### Training procedure

After having identified the list of items by using the aforementioned procedure, and in order to train the model, so as to recommend items given an input query, we are going to train the model using the following task. Given an input query, we first remove any stop words included in, then we identify the main entities (items) within this, and then we create training samples that include the pairs of queries (that do not contain the found item) and items. Figure 10 presents an example of the produced training samples based on a sample query.



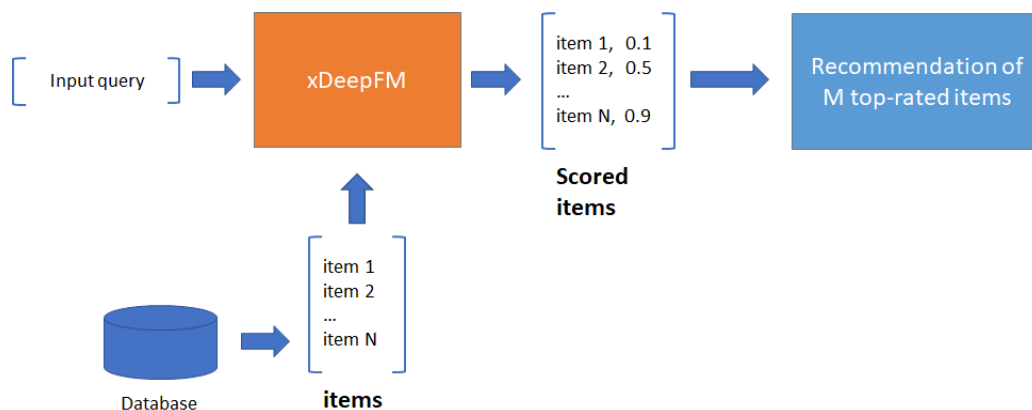
**Figure 10: Example of produced training samples for xDeepFM recommender based on a sample query**

The target for each training sample will be labeled as ‘1’ if the item was included in the initial sample query, while the rest of the items will be labeled as ‘0’. The training queries that do not contain tokens related to items will be filtered out.

Moreover, profile information of the users that performed the queries (e.g., gender, age group, interest, etc.) could be used during training, if such information is available, for providing more personalized queries recommendations.

### Inference procedure

For providing recommendation in real-time, the already trained extended xDeepFM model will be used. Given a user’s input query, the xDeepFM will calculate a score for each item that is within the system’s database. The M top-rated items will be used to augment the input query, and M new queries will be recommended to the users as relative queries. The exact number of the retrieved items will be defined after the evaluation of the algorithm and according to the needs of the EarthSearch module.



**Figure 11: Procedure for providing queries’ recommendations using the extended xDeepFM module**

#### 4.1.4 Platform’s Tools Software and Hardware Specifications

The following table demonstrates the software specifications of the Recommendation system.

**Table 10: Recommendation system’s tools specifications**

Licensing	A (non-open source) licensing of the development platform.
Core Implementation Technologies	Python, Flask
Additional technologies utilized	nlTK, pandas, scikit-learn, DeepCTR, tensorflow,
Database details	MySQL
Exposed APIs	RESTful web-services
Exchanged data format	JSON

#### 4.1.5 Exposed Services Specification

The Recommendation system exposes one REST web service that can be utilized by the rest of EarthSearch tools. The following table describes in detail the exposed by recommendation system service.

**Table 11: Description of the Exposed Services**

Service Info		
Service's Name	getSimilarQueries	
Description	Web service, which given an input query, the maximum number of responses to be retrieved and the method to be user, returns similar to the input query responses.	
Method	POST	
Url	<protocol>://<IP>:<port>/recommender/ getSimilarQueries	
Headers		
	Name	Value
	Accept	application/json, text/plain, */*
	Accept-Encoding	gzip, deflate, br

	Accept-Language	en-US,en;q=0.9	
Request Parameters			
	Name	type	Description
	Request's body	json	<p>The request's body consists of three parameters: a) the input query, b) the method that to be used by the recommendation module (i.e. sentence_similarity, xDeepFM) and c) the maximun number of similar queries to be returned.</p> <p>Example of request's body:</p> <pre>{   "input_query": "how to lose weight",   "method": "sentence_similarity",   "max_num_of_responses": 5 }</pre>
Request Example			
	POST	http://localhost:4200/ recommender/ getSimilarQueries	
Response			
	Status	Body	

	200	<p>Example of the response's body :</p> <pre>{   "input_query": "how to lose weight",   "similar_queries": [     "lose weight",     "ideal weight height ratio",     "the quickest way to lose weight",     "how much you should run to lose weight",     "how to lose weight fast"   ] }</pre>
--	-----	---

#### 4.1.6 Consumed Services and Usage Workflows

The recommendation system will not consume any web service provided by the rest of the modules of the platform. It will be only called by the search engine after the insertion of a user's query. The results of the recommendation system will be returned, as a list of similar queries, to the search engine, and through the search engine will be presented to the user. The described workflow is presented as a UML diagram in Figure 12.

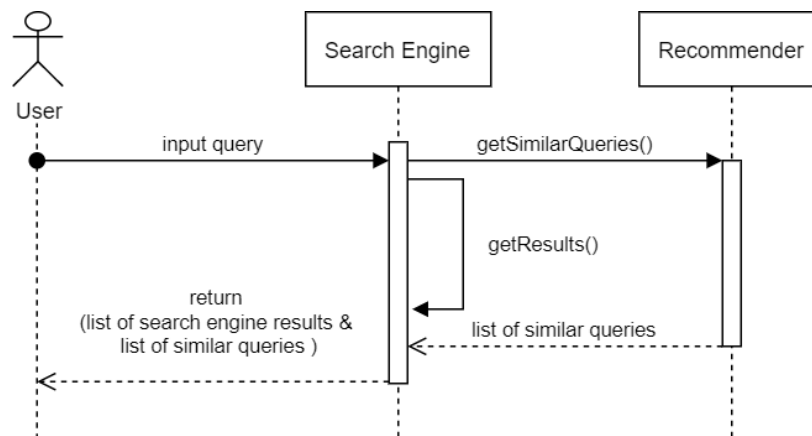


Figure 12: Recommendation system UML diagram – get similar queries

## 4.2 Accessibility scoring system

### 4.2.1 Introduction

Within the scope of WP5 and for addressing the objective 7 “Build an EO data access portal for the general public based on natural language processing and strongly integrated with the Qwant search engine” defined in the SnapEarth project’s Grant Agreement, an accessibility scoring system has been developed within Task 5.2. This system acts as an assistive module in the EarthSearch service, with the aim of evaluating the accessibility level of an image by taking into account special characteristics the users may have, such as vision impairments. The images provided as input in this system have resulted from a user’s search in Qwant’s web-search engine.

In particular, the accessibility scoring system aims to present a method for the evaluation of the accessibility level of an image using image filtering. The purpose of this system is to simulate how a user perceives an image under certain conditions, by applying the appropriate image filters, in order to measure how accessible that image is. For example, users with a visual impairment do not have the same perception on images as users with no visual impairments. This system has been designed to facilitate the accessibility of the images provided as output from the EO data portal. This system intends to enhance the accessibility in the images resulting from the web search engine without having any specific requirements input data.

This system will be integrated with Qwant’s web-search engine, allowing the user to select if s/he wants to receive the images resulting from a search query filtered and based on a certain type of filter. The input in this system can be either one single image or a list of images. The system will process these images according to the selected type of filter (e.g., visual impairment), and it will calculate for each one of them an accessibility score depicting how accessible an image is to the user, based on the selected type. According to this accessibility score, the images might be re-order and the images that are more easily accessible to the user, based on the selected condition, will be visualized first.

This accessibility scoring system is non-intrusive, can be used independently, and can be integrated into any other system. The image(s) can be provided as input in general by different sources than a web-search engine. Although, in the current case, the types of filters will be made available to the users through Qwant’s web-search engine interface. The communication of the accessibility scoring system and the rest modules of the EarthSearch platform will be established through the exposed web services of the accessibility scoring system, which will be described in detail in the next sections. This system will include types of filters concerning the accessibility levels allowed by the end-user devices, such as resizing images, and also type of filters for visually impaired users. The system’s evaluation method and results will be presented in D5.3.

### 4.2.2 Scope & Functional Description

The accessibility scoring system presented in this section aims at evaluating the accessibility level of the images provided as input based on image filtering. The image(s) provided as input may be the results from a web search engine (e.g., Qwant’s). The presented system focuses on the processing of images according to the type of filter selected by the user in order to provide an accessibility score for each image, depicting how accessible this image is, based on the characteristics of the selected filter.

The visual filters that will be applied for each type of filter have been defined according to their characteristics. The initial types of filters presented concern the visual impairment that a user might have. However, it should be noted that the methodology presented is general and can be extended to support additional types of filters.

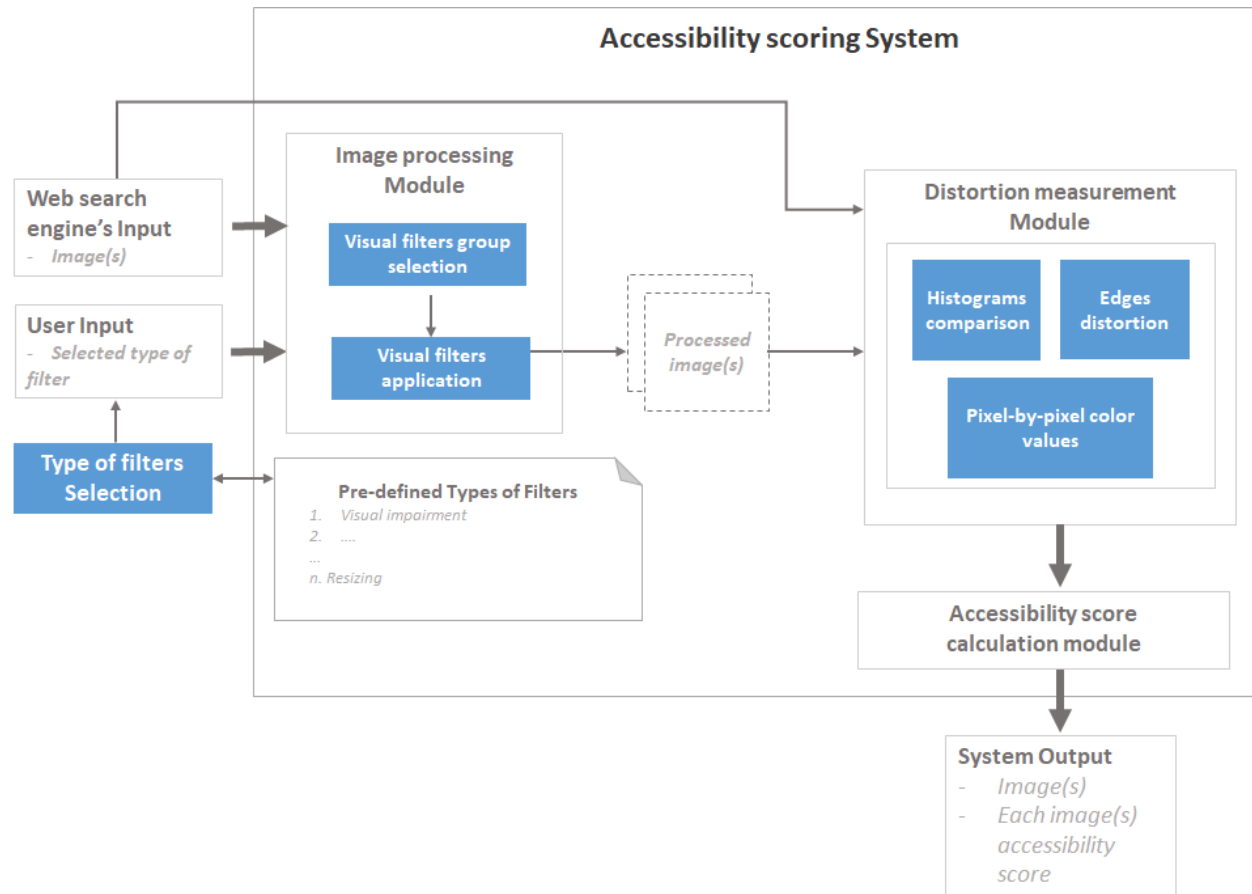
The main concept of this system is presented in Figure 13. The system receives as input an image or a list of images along with the type of filter. If a list of images is provided as input, each image will be processed independently, due to the nature of the implemented method. The type of filter parameter indicates the purpose under which the images should be evaluated (e.g., evaluate how accessible these images are for visually impaired people). It has been assumed that images with a higher accessibility score are more easily readable by the user. Each type of filter corresponds to a group of visual filters that will be applied in the input image, in order to simulate the users' visual perception and calculate accurately the accessibility level of each image. The types of filtering are predefined and will be available in a list, from which the users will be able to select the most suitable one according to their needs. These parameters are provided as input in the image processing module. The main aim of this module is initially to select the appropriate group of filters, based on the type of filter selected by the user, and then apply them to the input image(s).

Once the image has been processed and the appropriate filters have been applied, it is inserted as input along with the initial image in the distortion measurement module<sup>7</sup>. Within this module, the accessibility level of each image, called accessibility score, is calculated by measuring the distortion of the processed image, comparing it with the initial one. For the distortion measurement, three methods are used. The output of this module is a list, including the value resulting from each method applied.

For the calculation of the final accessibility score of each image, a corresponding module has been created. The output of the distortion measurement module is included in the final module, aiming to be used for the accessibility score's calculation. In this module, a weighted sum method takes place in order to achieve the final result. The final output is a list including the input images and the accessibility score calculated per image. This accessibility score can be used for the re-ordering of the images before they are provided to the end-user. This accessibility scoring system aims to be a valuable asset for all web search engines since it will increase user experience and user engagement.

---

<sup>7</sup> If the input is a list of images, the images are inserted one by one in the system.



**Figure 13: Accessibility scoring system's architecture**

In the next subsection, the methods included and applied within each module are presented. Evaluation results of this system will be presented in D5.3.

#### 4.2.3 Method Description

As described in Figure 13: Accessibility scoring system's architecture, the proposed system uses as input one or more images provided by Qwant's web-search engine and the type of filter selected by the user through Qwant's web-search engine's interface. The aim of this system is to process the inserted images and give as output an accessibility score for each one of them, according to the type of filtering selected by the user. The methods included in this system are based on the extraction of accessibility scores by comparing the input image and the image resulting after pre-processing, in order to measure its distortion.

The modules included in this procedure are the following:

- Image processing module:** processing of the input image(s) by applying the corresponding visual filters according to the type of filter selected.
- Distortion measurement module:** measures the distortion between the initial and the processed image.

- c. Accessibility score calculation: calculates the final accessibility level of the input image(s).

The methods implemented for each module are described below.

#### a) Image pre-processing module

This module concerns the processing of the input image(s) according to the type of filter selected. The input in this module requires two parameters, the first is an image or a list of images, and the second concerns the type of filter according to which the accessibility score will be calculated. Because each type of filter has a unique effect, different filters have to be applied during pre-processing in order to simulate the selected type's effects. Thus, in order to be able to simulate the user's perception for each type of filter, a group of visual filters that achieve the corresponding result has been created for each type of filter. According to the type of filter selected, the group of visual filters corresponding to this type are applied in the input image.

#### b) Measurement of the distortion between the initial and the pre-processed image

Calculation of the distortion measurement is the second module in the system's architecture, using as input the initial image and the processed one resulting from the previous step. During this step, three measurement functions are applied to measure the loss of information caused by filtering between the original and the filtered image. Both images are transformed from RGB to Luv color space before the application of each measurement method. The main aim of using these three methods is to provide a valuable result for the distortion measurement.

These methods are:

- **Histograms comparison:** In this function, only the L channel of both processed images is used. The luminance histograms of both the original and the filtered image are extracted and compared. The histograms' difference distortion function is then calculated using the Euclidean distance between the histograms of the two images.

$$r_h(I, I_m) = \sqrt{\sum_{u=1}^b (h_{I,k} - h_{I_m,k})^2} \quad (5)$$

Where  $h$  is the histogram of the input image  $I$  and the processed image  $I_m$  respectively,  $u=1, \dots, b$  is the value of the  $u$ -th bin in both the input and the processed images.

- **Edges distortion:** For this function, the L channel of both the original and the filtered image is used as input. The Sobel edge detection [55] operator is applied to both images in order to mark their edges. The new images produced after the application of the Sobel edge detection operator are in greyscale. Each pixel's value is normalized to the  $[0, 1]$  in order to define if it is an edge or not. The closer the intensity is to 1, the sharper an edge exists in the respective initial image.

Aiming at calculating one single value from this method, which will be further used for the calculation of the final accessibility score, the number of edges (a) included for both the processed and the original input mages is calculated. This number of edges is calculated for each pixel in both the initial and the processed images using the following equation:

$$a_I = \frac{1}{hw} \sum_{i=1}^h \sum_{j=1}^w n^{ij} \quad (6)$$

Where I is the input image provided, either the processed one or the initial one, the h is the height of the image, the w is the width of the image and the n variable is the intensity of the i-th row of the image, and j-th column.

The final edges distortion value is calculated using the following equation:

$$e(I, I_m) = \frac{a_I - a_{I_m}}{a_I} \quad (7)$$

Where I is the initial image and  $I_m$  is the processed image.

- **Pixel-by-pixel color values:** This function calculates the pixel-by-pixel difference in color between the original and the filtered images. For this method, the squared color difference between pixels (d) in the (i, j) position of the pixels in each image. In both images, the R, G, B color components of each pixel are retrieved. This squared difference is measured as follows:

$$d_{ij}^2 = (I^{ij,R} - I_m^{ij,R})^2 + (I^{ij,G} - I_m^{ij,G})^2 + (I^{ij,B} - I_m^{ij,B})^2 \quad (8)$$

where I is the initial image and  $I_m$  is the processed image.

In order to be able to translate the difference of the pixels between the two images and ignore small differences in color that could exist due to noise, a threshold is added. According to this threshold, the value of the pixel's difference ( $t_{ij}$ ) is 1 if the  $d_{ij}^2$  is greater than the threshold set, while in the rest cases this value is 0. For the calculation of the final value of distortion, the following equation is used:

$$r_p(I, I_m) = \frac{1}{hw} \frac{1}{C} \sum_{i=1}^h \sum_{j=1}^w t_{ij} d_{ij}^2 \quad (9)$$

This equation is calculated for both the input image I and the processed image  $I_m$ . The parameter C is a normalization constant calculated by the following equation:

$$C = \sum_{i=1}^h \sum_{j=1}^w t_{ij} \quad (10)$$

### c) Accessibility score calculation module

The outcome of each distortion method provided by the previous module is combined on a weighted sum in order to calculate the total distortion of the original image. This result is a single value in between [0, 1]. A linear combination of the three distortion measures has been used due to its simplicity and capability of assigning significance weights to the individual distortion measures. The equation followed for the calculation of the accessibility score of each of the input images is:

$$s_{k,m} = w_h r_h(I, I_m) + w_e e(I, I_m) + w_p r_p(I, I_m) \quad (11)$$

where k is the index of the input image provided, m is the input image,  $w_h$  is the weight set by the system for the histogram comparison method, the  $w_e$  is the weight set by the system for the edges detection method and,  $w_p$  is the weight set by the system for the pixel-by-pixel color value method.

#### 4.2.4 Platform’s Tools Software and Hardware Specifications

The following table demonstrates the software specifications of the Accessibility scoring system.

**Table 12: Accessibility scoring system’s specifications**

Licensing	A (non-open source) licensing of the development platform.
Core Implementation Technologies	Python.
Additional technologies utilised	pandas, scikit-learn, numpy, opencv, pythologist-image-utilities
Exposed APIs	RESTful web-services
Exchanged data format	JSON

#### 4.2.5 Exposed Services Specification

The accessibility scoring system exposes one REST web service that can be utilized by the rest of EarthSearch tools. The following table describes in detail the exposed the accessibility scoring system service.

**Table 13: Description of the Exposed Services**

Service Info			
Service's Name	imgsAccessibility		
Description	Web service, which given as input a list of images and information about the filter selected by the user, returns an accessibility score presenting how accessible a given image is based on the attributes of the selected filter.		
Method	POST		
Url	<protocol>://<IP>:<port>/accessibility/imgsAccessibility		
Headers			
	Name	Value	
	Accept	application/json, text/plain, */*	
	Accept-Encoding	gzip, deflate, br	
	Accept-Language	en-US,en;q=0.9	
Request Parameters			
	Name	type	Description
	Request's body	json	<p>The request's body consists of two parameters: a) a list of the images, including their name and the link to be retrieved from and b) the type of filter selected by the user.</p> <p>Example of request's body:</p> <pre>{  "images":[</pre>

			<pre>{   "img_link": "",   "img_name": "" },  {   "img_link": "",   "img_name": "" } ],  "filter_selected": "visual_impairment" }</pre>
Request Example			
	POST	http:// <IP>:<port>/accessibility/imgsAccessibility	
Response			
	Status	Body	

	200	<p>Example of the response's body :</p> <pre> “res”:[{   "score": 1,   "img_name": "" }, {   "score": 0.5,   "img_name": "" }, {   "score": 0.9,   "img_name": "" }] </pre>
--	-----	---

#### 4.2.6 Consumed Services and Usage Workflows

The accessibility scoring system will not consume any web service provided by the rest modules of the platform. It will exchange information with the web search engine, where this module will be integrated, through which the users can select to evaluate the accessibility level of images resulting from a query according to a certain filter. The results of the accessibility scoring system will return a list including the accessibility score of each image, based on the type of filter selected, and will be available to the end-user. The described workflow is presented as a UML diagram in Figure 14.

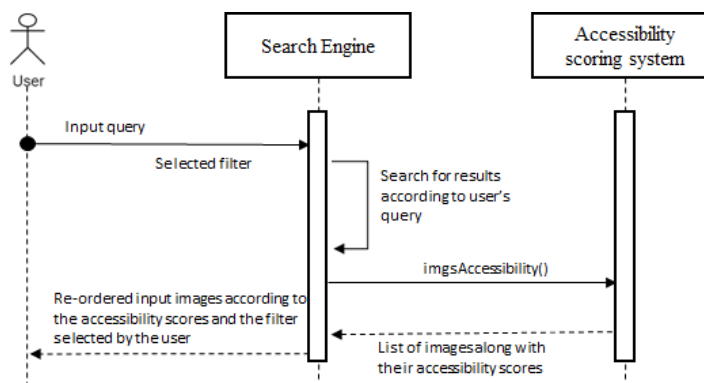


Figure 14: Accessibility scoring system's UML diagram.

## 5. Conclusion

In this document, we present the overall architecture of the EarthSearch service. We survey our progress on the development of the API of the service, which feature we implemented in the first period of the project, and which feature we plan to implement in the second part of the project. We describe extensively the principles of the VQA system we implement, and the services exposed. We also outline our accessibility scoring system, its function, and how we will use it in the project.

## Bibliography

- [1] Giakoumis D, Kaklanis N, Votis K, Tzovaras D (2013) Enabling user interface developers to experience accessibility limitations through visual, hearing, physical and cognitive impairment simulation, pp 1–22. Universal Access in the Information Society
- [2] National eye institute. <https://nei.nih.gov> (2015)
- [3] Brettel H, Vi'enot F, Mollon JD (1997) Computerized simulation of color appearance for dichromats. *JOSA A* 14(10):2647–2655
- [4] Wang, H., Zhang, F., Xie, X., & Guo, M. (2018, April). DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference* (pp. 1835-1844).
- [5] Thorat, P. B., Goudar, R. M., & Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4), 31-36.
- [6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3146–3154.
- [7] Khan, B. M., Mansha, A., Khan, F. H., & Bashir, S. (2017, December). Collaborative filtering based online recommendation systems: A survey. In *2017 International Conference on Information and Communication Technologies (ICICT)* (pp. 125-130). IEEE.
- [8] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173-182).
- [9] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020, July). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 639-648).
- [10] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.
- [11] Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, H. (2016, September). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7-10).
- [12] Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*.

- [13] Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. (2018, July). xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1754-1763).
- [14] Coello CAC, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multiobjective problems, vol 5. Springer
- [15] Ehrgott M (2005) Multicriteria optimization, vol 2. Springer, Berlin
- [16] Manning CD, Raghavan P, Hinrich S et al (2008) Introduction to information retrieval, vol 1. Cambridge University Press, Cambridge
- [17] Zitzler E, Laumanns M, Thiele L, Zitzler E, Zitzler E, Thiele L, Thiele L (2001) SPEA2: improving the strength pareto evolutionary algorithm
- [18] Nikulin Y, Miettinen K, Mäkelä MM (2012) A new achievement scalarizing function based on parameterization in multiobjective optimization. OR Spect 34(1):69–87
- [19] Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning (Vol. 242, No. 1, pp. 29-48).
- [20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.
- [21] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546.
- [22] word2vec. (2021, 02, 04). code.google.com. <https://code.google.com/archive/p/word2vec/>
- [23] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).
- [24] GloVe: Global Vectors for Word Representation. (2021, 02, 04). nlp.stanford.edu, <https://nlp.stanford.edu/projects/glove/>
- [25] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- [26] Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text classification. In International conference on intelligent data engineering and automated learning (pp. 611-618). Springer, Berlin, Heidelberg.
- [27] Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. (2018, July). xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1754-1763).

- [28] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision* (pp. 19-27).
- [29] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- [30] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
- [31] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). VQA: Visual question answering. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter, 2425–2433. <https://doi.org/10.1109/ICCV.2015.279>
- [32] Audebert, N., Le Saux, B., & Lefèvre, S. (2018). Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140, 20–32. <https://doi.org/10.1016/j.isprsjprs.2017.11.011>
- [33] Badrinarayanan, V., Handa, A., & Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. <http://arxiv.org/abs/1505.07293>
- [34] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), 2481–2495.
- [35] Gupta, Ritwik, Hosfelt, R., Sajeev, S., Patel, N., Goodman, B., Doshi, J., Heim, E., Choset, H., & Gaston, M. (2019). XBD: A dataset for assessing building damage from satellite imagery. *ArXiv*, 10–17.
- [36] Gupta, Rohit, & Shah, M. (2020). RescueNet: Joint building segmentation and damage assessment from satellite imagery. *ArXiv*.
- [37] Jing, W., Ren, Q., Zhou, J., & Song, H. (2020). AutoRSISC: Automatic design of neural architecture for remote sensing image scene classification. *Pattern Recognition Letters*, 140, 186–192. <https://doi.org/10.1016/j.patrec.2020.09.034>
- [38] Kafle, K., & Kanan, C. (2017). Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163, 3–20. <https://doi.org/10.1016/j.cviu.2017.06.005>
- [39] Kordelas, G. A., Manakos, I., & Bustamante, J. (n.d.). Fast and Automatic Data-Driven Thresholding for Inundation Mapping with Sentinel-2 Data. 1–23. <https://doi.org/10.3390/rs10060910>
- [40] Li, W., He, C., Fang, J., Zheng, J., Fu, H., & Yu, L. (2019). Semantic segmentation-based building footprint extraction using very high-resolution satellite images and multi-source GIS data. *Remote Sensing*, 11(4). <https://doi.org/10.3390/rs11040403>

- [41] Lobry, S., Marcos, D., Kellenberger, B., & Tuia, D. (2020). Better Generic Objects Counting When Asking Questions to Images: A Multitask Approach for Remote Sensing Visual Question Answering. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5(2), 1021–1027. <https://doi.org/10.5194/isprs-annals-V-2-2020-1021-2020>
- [42] Lobry, S., Marcos, D., Murray, J., & Tuia, D. (2020). Rsvqa: Visual question answering for remote sensing data. *ArXiv*, 58(12), 8555–8566.
- [43] Lobry, S., & Tuia, D. (2019). Deep learning models to count buildings in high-resolution overhead images. 2019 Joint Urban Remote Sensing Event, JURSE 2019, 2019–2022. <https://doi.org/10.1109/JURSE.2019.8809058>
- [44] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- [45] Open Access Hub. (n.d.-a). Retrieved March 25, 2021, from <https://scihub.copernicus.eu/userguide/>
- [46] Open Access Hub. (n.d.-b). Retrieved March 25, 2021, from <https://scihub.copernicus.eu/>
- [47] Sentinel-2 - Missions - Sentinel Online - Sentinel. (n.d.). Retrieved March 28, 2021, from <https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-2>
- [48] What is Copernicus | COPERNICUS EMERGENCY MANAGEMENT SERVICE. (n.d.). Retrieved March 25, 2021, from <https://emergency.copernicus.eu/mapping/ems/what-copernicus>
- [49] What is Remote Sensing? | Earthdata. (n.d.). Retrieved March 25, 2021, from <https://earthdata.nasa.gov/learn/backgrounders/remote-sensing#data-processing-interpretation-and-analysis>
- [50] What is remote sensing and what is it used for? (n.d.). Retrieved March 28, 2021, from [https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used?qt-news\\_science\\_products=0#qt-news\\_science\\_products](https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used?qt-news_science_products=0#qt-news_science_products)
- [51] Wheeler, B. J., & Karimi, H. A. (2020). Deep learning-enabled semantic inference of individual building damage magnitude from satellite images. *Algorithms*, 13(8). <https://doi.org/10.3390/A13080195>
- [52] MovieLens. (2021, 02, 04). grouplens.org. <https://grouplens.org/datasets/movielens/>
- [53] Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, H. (2016, September). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7-10).
- [54] Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247*
- [55] Gao, W., Zhang, X., Yang, L., & Liu, H. (2010, July). An improved Sobel edge detection. In *2010 3rd International conference on computer science and information technology* (Vol. 5, pp. 67-71). IEEE.

- [56] Sen2-Agri-Product-Cropland-Mask document. Retrieved from <http://www.esa-sen2agri.org/wp-content/uploads/resources/technical-documents/Sen2-Agri-Product-Cropland-Mask.pdf>
- [57] Sen2-Agri-Product-CropType-Map document. Retrieved from <http://www.esa-sen2agri.org/wp-content/uploads/resources/technical-documents/Sen2-Agri-Product-Crop-Type-Map.pdf>
- [58] Sen2-Agri-Product-Vegetation-Status document. Retrieved from <http://www.esa-sen2agri.org/wp-content/uploads/resources/technical-documents/Sen2-Agri-Product-Vegetation-Status.pdf>