Horizon 2020

Understanding Europe's Fashion Data Universe

Showcase Specification

Deliverable number: D7.4

Version 3.0

Funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 732328

Project Acronym: FashionBrain
Project Full Title: Understanding Europe's Fashion Data Universe
Call: H2020-ICT-2016-1
Topic: ICT-14-2016-2017, Big Data PPP: Cross-sectorial and cross-lingual data integration and experimentation
Project URL: https://fashionbrain-project.eu

Deliverable type	Report (R)
Dissemination level	Public (PU)
Contractual Delivery Date	30 June 2017
Resubmission Delivery Date	28 February 2019
Number of pages	20, the last one being no. 14
Authors	Alessandro Checco - USFD Matthias Dantone - Fashwell
Peer review	Jennifer Dick - USFD

Change Log

Version	Date	Status	Partner	Remarks
1.0	03/07/2017	Final	USFD	Rejected 15/03/2018
2.0	20/04/2018	Resubmitted Final	USFD	Rejected 15/10/2018
3.0	28/02/2019	Resubmitted Final	USFD	

Deliverable Description

This deliverable contains a specification of the FashionBrain data integration infrastructure including design of promotion material and requirements for software needed to run it.

Abstract

The FashionBrain project provides solutions for data integration in the fashion industry.

In month 36, a showcase will be provided to demonstrate the FashionBrain data integration infrastructure with interactive demos and screencasts to the public. This showcase can be used by the Commission for its own dissemination and promotional activities (including Web based and electronic publications) after the completion of the project.

This deliverable reports the functional and technical specifications of the different parts of the showcase, explaining how to access them and which software is needed to run them.

Table of Contents

List of Figures

List of Acronyms and Abbreviations

API	Application Programming Interface
CD	Centroid Decomposition
CQE	Continuous Query Engine
CSV	Comma Separated Values
DSSM	Deep Structured Semantic Models
FaBIAM	FashionBrain Integrated Architecture
IDEL	In-Database Entity Linking
IoT	Internet of Things
JPG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
PDF	Portable Document Format
RDBMS	Relational Database Management System
REST	REpresentational State Transfer
SQL	Structured Query Language
UDF	User-Defined Function

1 Introduction

The FashionBrain project provides solutions for data integration in the fashion industry. This deliverable will lay the groundwork for the set of demonstrations that will be used to showcase the project.

The main solutions we will showcase are:

Shop the Look. We demonstrate the core functionalities that allow FashionBrain to detect and recognize fashion items like t-shirts, bags, etc., from images, and return the most similar products from a specific data set.

End-to-end Search. We demonstrate the FashionBrain ability to match textual search queries to a ranked list of fashion products, by using a mixture of computer vision methods and Natural Language Processing (NLP) to create a smart search index, able to cope with complex queries and multilinguality.

FashionBrain Integrated Architecture (FaBIAM). We provide a showcase and documentation of FaBIAM, a MonetDB-based architecture for storing, managing and analysing of both structured and unstructured data. This architecture allows the implementation of in-database analytical solutions with Machine Learning (ML).

Entity Recognition and Relation Extraction. We demonstrate how complex fashion corpora (e.g., fashion blogs and news) can be automatically analysed and annotated by our tools.

A set of demonstrators and promotional documents will be packaged by the end of the project (M36). These can be used by the Commission for its own dissemination and promotional activities (including web based and electronic publications) after the completion of the project.

The showcase is an evolution of the demos presented during the project, with the notable difference that in this form, the solutions presented are ready to be distributed in an interactive or static form (e.g. web sites, videos or documents) and consumed by a layman, without the need of a technical demonstrator nor a deep understanding of the input required.

1.1 Scope of This Deliverable

This deliverable is a precursor to D7.6, that will contain extended information about the showcase and promotional material. Here, we focus on the **architecture design** and the **software required** to run the demonstrators.

For more information about the details of each part of the showcase, we refer to: (i) D4.2, D4.4 and D2.1 for entity recognition and relation extraction; (ii) D6.3 and D6.5 for end-to-end search; (iii) D5.2 for image recognition; (iv) D2.3 and D2.4 for FaBIAM and time series operators in MonetDB.

1.2 Showcase Structure

The showcase will be available on the FashionBrain website[1]. All of the datasets needed to run the demos will be hosted in the cloud as further explained in the next section. Moreover, the solutions showcased will be presented in an interactive form, a static form (suitable for printed publication) and a video form.

For each part of the data-integration pipeline, we will provide a specific section in the project website that will contain: (i) a description of the dataset used; (ii) an explanation of the data integration techniques performed on the dataset and the underlying architecture; (iii) a demo of the techniques described.

Reliability and Maintenance

The FashionBrain data integration solutions require access to high performance computers and complex interaction between multiple services. Providing a long lasting offline solution to the Commission would be impractical, as it would require the duplication of a very expensive infrastructure.

For this reason, and to guarantee a reliable access to the promotional material, we will provide an interactive version of the solutions (as explained in detail in the *Specification* section of each showcase in the following), pre-applied on a meaningful subset of the data used during the project, as well as a set of static screencasts and webpages that can be reused by the Commission to promote FashionBrain solutions offline.

This solution will guarantee the maximisation of reliability and the minimisation of the effort needed to maintain the information provided after the end of the project.

[1]Instructions to host the showcase in other websites will be provided in D7.6.

2 Showcase 1 – Shop the Look

In this showcase we demonstrate the data integration process via the “search by image” functionality. The goal is to demonstrate the ability to build an application able to detect and recognize fashion items like t-shirts and bags from images, and return a ranked listed of similar product from a dataset, as shown in Figure 2.1.

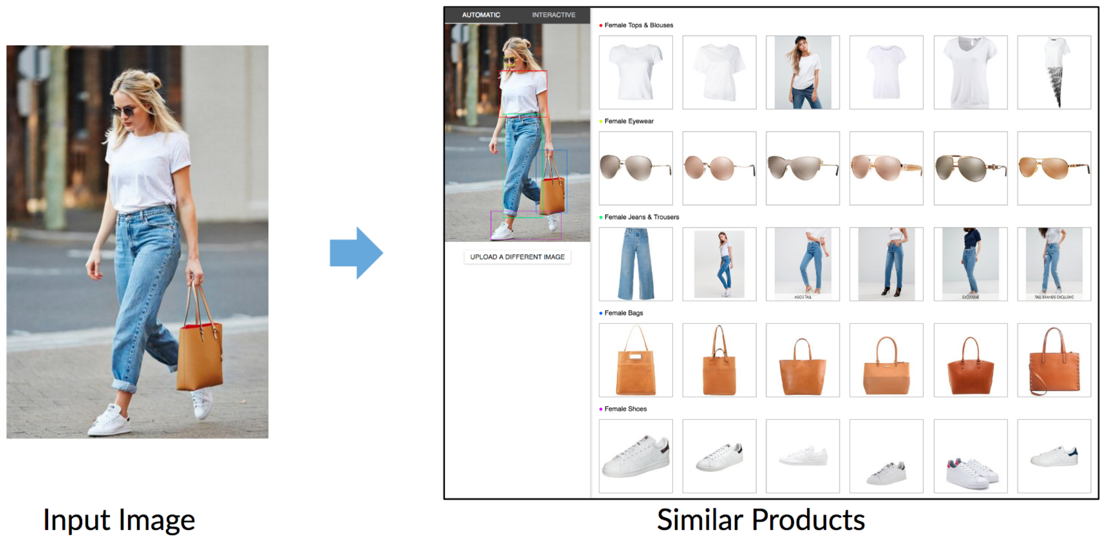


Figure 2.1: Shop the look Interface.

2.1 Architecture

The client-server architecture consists of a website and a cloud deployed recognition service. The client sends the images to this recognition service, which in turn localizes and recognizes products in the uploaded images and returns the most similar products from the pre-defined product database. The services required are deployed as stateless micro-services. This ensures scalability and also allows one to develop the individual components independently. All services are packed into docker containers and deployed using Kubernetes, which takes care of node scheduling, service discovery, load balancing and deployment / rollback. Each component has its own load balancer and can be scaled individually.

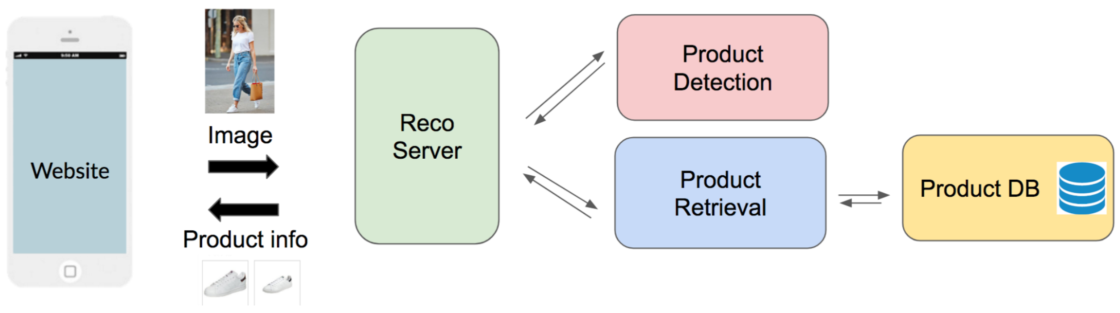
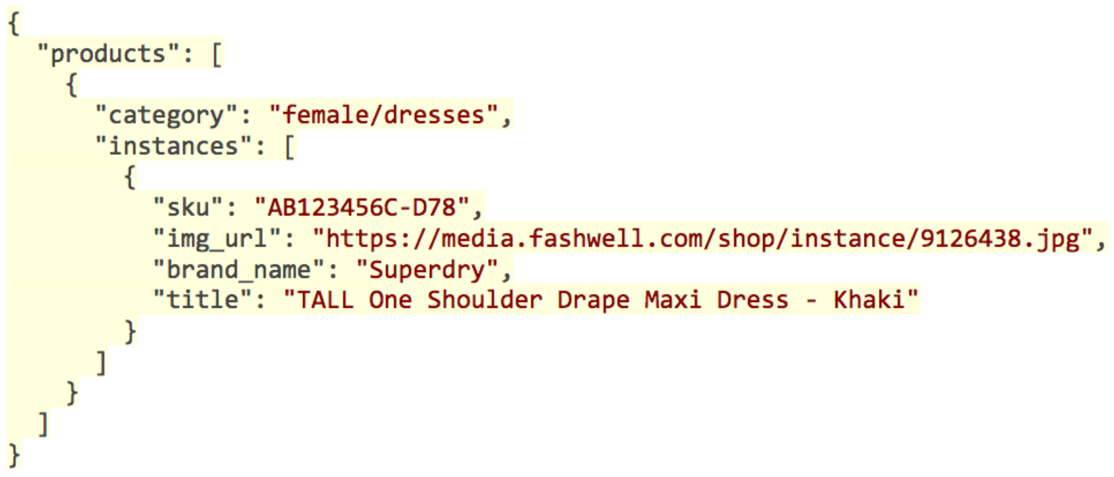


Figure 2.2: Shop-the-look architecture.

Recognition Server

This service implements the API. It receives requests from the client via REpresentational State Transfer (REST), validates them and forwards them to the different upstream services. A request consists of a single three channel JPG image with minimum dimension of 300x300 pixel and a max dimension of 1000x1000 pixel, compressed with JPG quality 30 (`https://arxiv.org/abs/1604.04004`). Also, the request contains a device identifier or a session identifier.

This service responds for each detected object, normalized bounding box coordinates with its fashion category and a short description of every found product including the similar products from the database. A sample response could look like Figure 2.3. Fashwell is going to provide an API for the recognition service.



```
{
  "products": [
    {
      "category": "female/dresses",
      "instances": [
        {
          "sku": "AB123456C-D78",
          "img_url": "https://media.fashwell.com/shop/instance/9126438.jpg",
          "brand_name": "Superdry",
          "title": "TALL One Shoulder Drape Maxi Dress - Khaki"
        }
      ]
    }
  ]
}
```

Figure 2.3: Shop the look API structure.

Product Detection

For localization, we group all fashion products into basic semantic classes, example: bags, shoes, jackets & coats. By doing so, detection will be much more robust, since the core categories are visually very different. This service receives a single image and returns bounding box coordinates which are normalized with respect to the image size with the recognized core category and a confidence score.

Product Retrieval

The product retrieval gets as an input the image patch with the detected product and returns a list of similar products. In order to make that system possible the

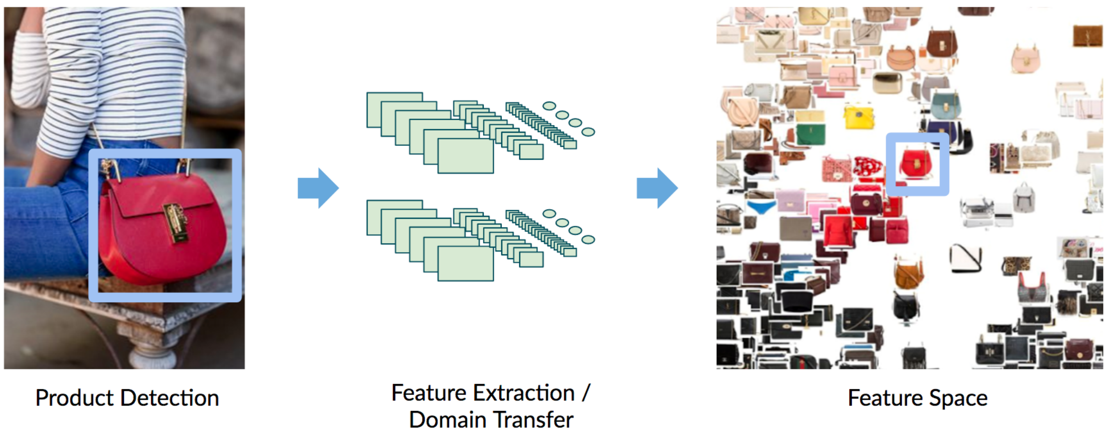


Figure 2.4: Shop the look domain transfer.

consortium needs to build two systems: Feature Extraction and a Visual Index. The visual index is a feature space where the all products are present and can be queries. That space has the requirement that similar products lie closer together than not similar products. The feature extraction step makes the transformation from the the image space to the visual index space.

Product Database

Building the product database is a challenging task for the consortium. Fashwell is providing the relevant product catalog for this task. The product catalog is going to have all products from around 100-200 online shops from different countries. The different products catalog have different formats and metadata structure. For this reason a product feed normalization step is needed (see Deliverable D1.3 for more information on the FashionBrain taxonomy). The final output is a visual index, that is a searchable version of the unified product feed.

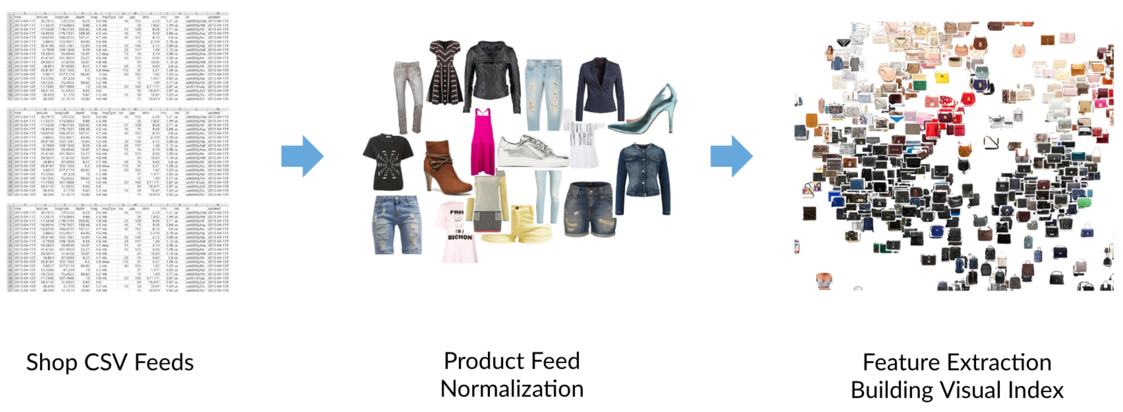

Shop CSV Feeds
Product Feed Normalization
Feature Extraction Building Visual Index

Figure 2.5: Shop the look pipeline.

2.2 Specification

This demonstration will contain a meaningful subset of images. The user will be able to select an image and visualise the most similar product present in the FashionBrain dataset, together with a description of the inferred position in the FashionBrain taxonomy.

Figure 2.6 shows an example of such an interface.

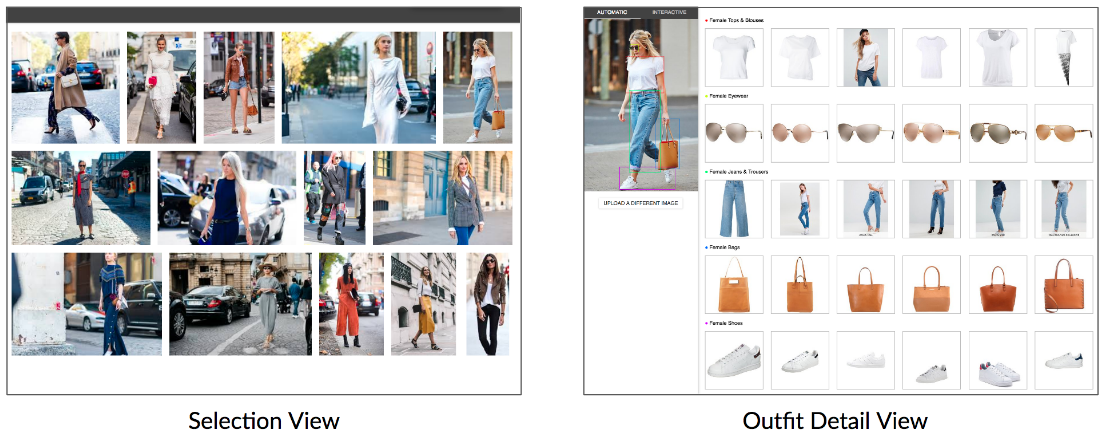


Figure 2.6: Shop the look mock-up.

Additionally, a printed-friendly document and a screencast will be provided.

2.2.1 Software Requirements

The only software required is a modern browser and a PDF reader.

3 Showcase 2 - End-to-end Search

This showcase will demonstrate the end-to-end search capabilities achieved during the development of the FashionBrain project.

The main goal is to match textual search queries to a ranked list of fashion products, by using a mixture of computer vision methods and NLP to create a smart search index, able to cope with complex queries and multilinguality.

3.1 Architecture

We embed images and text into a shared vector space that model multimodal text-image semantic similarity. To achieve this, we adopt an architecture based on Deep Structured Semantic Models (DSSM) and trained with a rank-loss hinge objective. This architecture defines two separate neural networks, one for each data type: The first, a recurrent neural network, is used to process text, while the second, a convolutional neural network, is used to process images. Both networks are trained to produce embedding vectors that are similar according to the cosine distance if a text matches an image. An overview of this architecture is shown in Figure 3.1.

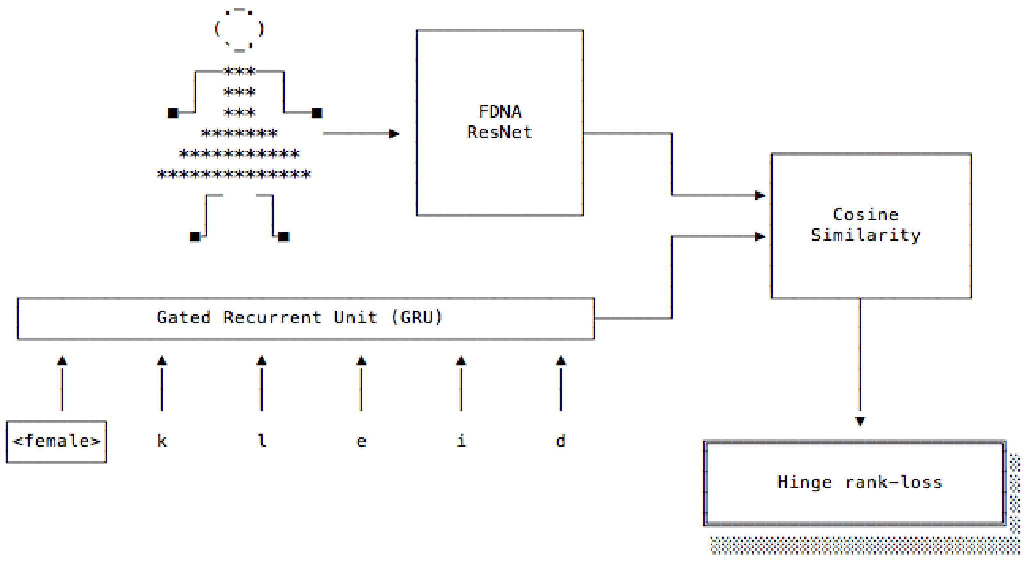
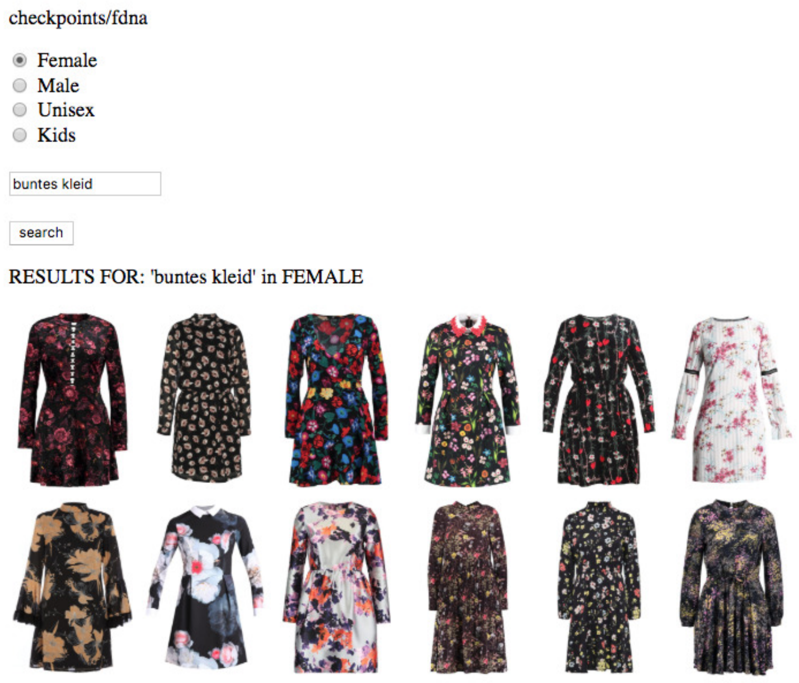


Figure 3.1: Illustration of proposed neural information retrieval approach.

We refer to Deliverables D6.3 and D6.5 for more details on the architecture.

3.2 Specifications


checkpoints/fdna

Female [✓]
Male
Unisex
Kids

buntes kleid

search

RESULTS FOR: 'buntes kleid' in FEMALE

Figure 3.2: Search results for "buntes Kleid", which is "colorful dress" in English.

This demonstration will contain a meaningful subset of queries in multiple languages. The user will be able to select a query and visualise a ranked set of products from the FashionBrain dataset.

Figure 3.2 shows an example of such an interface.

Additionally, a printed-friendly document and a screencast will be provided.

3.2.1 Software Requirements

The only software required is a modern browser and a PDF reader.

4 Showcase 3 – FashionBrain Integrated Architecture

In this showcase, we detail the design and implementation of FashionBrain Integrated Architecture (FaBIAM), a MonetDB-based architecture for storing, managing and analysing of both structured and unstructured data. The showcase will provide examples of novel ML solutions that can be implemented in-database. We refer to Deliverable D2.3 for more details on the underlying infrastructure.

4.1 Architecture

Figure 4.1 shows an overview of FaBIAM. All components are integrated into the kernel of MonetDB. The solid arrows indicate the components that can already work together, while the dashed arrows indicating future integration. From bottom to top, they are divided into three layers:

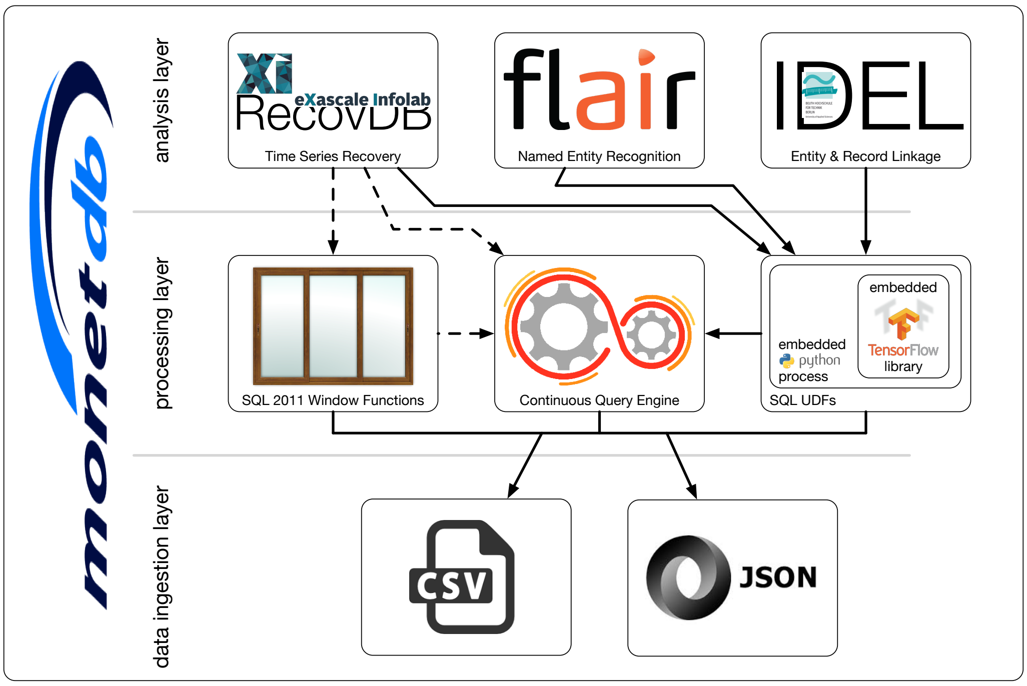


Figure 4.1: Architecture of the FashionBrain integrated architecture (FaBIAM).

Data Ingestion Layer This layer at the bottom of the MonetDB kernel provides various features for loading data into MonetDB. In the fashion world, there are three major groups of data: structured (e.g. product catalogues and sales information), unstructured (e.g. fashion blogs, customer reviews, social media posts and news messages) and binary data (e.g. videos and pictures). A prerequisite for the design of FaBIAM is that it must be able to store and process both structured and unstructured data, while binary data can be generally left as is. Therefore, next to Comma Separated Values (CSV) (the de facto standard data format for structured data) MonetDB also support JavaScript Object Notation (JSON) (the de facto standard data format for unstructured data) as a native data type.

Processing Layer This layer in the middle of the MonetDB kernel provides various features to facilitate query processing. In the context of the FashionBrain project, we have introduced several major extensions in this layer geared towards streaming and time series (fashion) data processing by means of both traditional SQL queries, as well as using modern machine learning technologies. This include i) major extensions to MonetDB's support for Window Function; ii) a Continuous Query Engine (CQE) for streaming and Internet of Things (IoT) data; and iii) a tight integration with various machine learning libraries, including the popular TensorFlow library, through Structured Query Language (SQL) Python User-Defined Function (UDF)s.

Analysis Layer In this layer at the top of the MonetDB kernel, we have integrated technologies of FashionBrain partners (under the collaborations of the respective partner) to enrich MonetDB's analytical features for (fashion) text data and time series data:

- *FLAIR* [1] is a python library (provided by Zalando) for named entity recognition.
- *In-Database Entity Linking (IDEL)* [3] is also a python library (provided by BEUTH), but for linking of already identified entities between text data and relational records, and for records linkage of already identified entities in relational records.
- *RecovDB* [2] is a MonetDB-based Relational Database Management System (RDBMS) for the recovery of blocks of missing values in time series stored in MonetDB. The Centroid Decomposition (CD)-based recovery algorithm (provided by UNIFR) is implemented as SQL Python UDFs, but UNIFR and MDBS are working together on porting it to MonetDB native C-UDFs.

In summary, the design of the FaBIAM architecture covers the whole stack of data loading, processing and analysis specially for fashion text and time series data.

4.2 Specification

The demonstration will contains the following examples:

1. We will detail how JSON data can be loaded into MonetDB and queried.

2. We will describe how one can use FLAIR from within MonetDB through SQL Python UDF.
3. We will demonstrate how FaBIAM can be used to process, analyse and store a stream of reviews posted by Zalando customers.

The users will be able to select different ML examples, and visualise a set of instructions to implement them in MonetDB.

Additionally, a print-friendly documentation will be provided.

4.2.1 Software Requirements

The only software required to visualise the examples of the showcase is a modern browser and a PDF reader. Additionally, MonetDB[1] and all the software containing the architecture will be downloadable from, e.g., Github.

[1]Open source: https://www.monetdb.org/Downloads.

5 Taxonomy Enrichment and Relation Extraction

In this showcase we will demonstrate how the FashionBrain technology is able to automatically analyse complex text from a corpus of fashion blogs, articles, news reports, to provide:

1. Entity linking and mapping to a fashion taxonomy (Taxonomy Enrichment).
2. Automatic complex relations extraction.

We refer to deliverables D4.2, D4.4 and D2.1 for more information on the entity recognition and relation extraction architecture.

5.1 Architecture

Entity Recognition

Taxonomies give a clear and easy representation to understand a knowledge base. However, fashion taxonomies are either inconsistent or have missing items. These missing items need to be integrated to our pre-built taxonomy in order to enrich it. Figure 5.1 summarises the pipeline of our solution. We first start with the

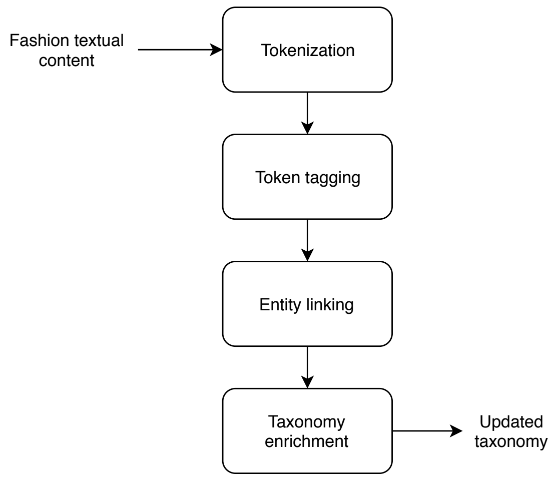


Figure 5.1: Taxonomy Enrichment.

Named Entity Recognition (NER) step. This step consists in i) tokenizing the textual content, tagging it, i.e., each word is identified as a noun, a verb, an adjective or a date and ii) identifying entities in predefined categories such as persons, organizations, locations, etc. We apply CoreNLP on textual to perform the NER step (see Deliverable D2.1 for more information).

Relation Extraction

Insights in fashion related events are often desired by internal stakeholders, such as specialists working in strategic areas, marketing departments or in supply chain management/logistics. These specialists seek insights for optimizing business processes or for reacting on recently appearing events. In brand monitoring, stakeholders in Zalando desire to monitor brands, products, but also events such as product recalls, new product announcements, acquisitions, company suppliers, company competitors, man made disasters, mergers and company customers. Extracting complex relations between fashion items, influencers, and events is fundamental to correctly understand the market.

The fashion domain has a high turnover; often brands and products might appear every two months. Deep learning based methods generalize well, and can be stacked into each other and with shallow learners. The result is a learning framework that benefits from high generalization capabilities of deep learning models for basic language specific text mining tasks (such as recognizing entities) into models for domain specific tasks, such as linking entities or recognizing relations.

We will learn n-ary relation extraction functions for fashion related events. Our methods will execute the process of recognizing relations and entities jointly and will leverage LSTMs and other recurrent neural networks.

In contrast to other methods, this approach will jointly learn to identify the entities pertaining to a relation, as well as the type of relation itself. It does so utilizing multi step hierarchical reinforcement learning on top of a regular supervised learning scheme. In the first step the model determines the type of relation that is contained in a given context. Using that information a subroutine now aims to select the entities belonging to that relation type only. This simplifies the problem of named entity recognition, as this method is akin to learning separate entity recognizers for each different relation type within a single model instead of having to learn a generic entity recognizer.

5.2 Specification

This demonstration will contain a meaningful subset of sentences from a fashion corpus. The user will be able to select a sentence and visualise the recognised taxonomy item, and, if present, the relation between different entities.

Additionally, a printed-friendly document and a screencast will be provided.

5.2.1 Software Requirements

The only software required is a modern browser and a PDF reader.

Bibliography

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[2] Ines Arous, Mourad Khayati, Philippe Cudré-Mauroux, Ying Zhang, Martin Kersten, and Svetlin Stalinlov. RecoveDB: accurate and efficient missing blocks recovery for large time series. In *Proceedings of the 35th IEEE International Conference on Data Engineering (ICDE 2019)*, April 2019. Submitted.

[3] Torsten Kilias, Alexander Löser, Felix Gers, Ying Zhang, Richard Koopmanschap, and Martin Kersten. IDEL: In-Database Neural Entity Linking. In *Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feburary 2019. To appear.