
Horizon 2020 Framework Programme

Grant Agreement: 732328 – FashionBrain

# Document Information

**Deliverable number:** D6.3

**Deliverable title:** Early demo on textual image search

**Deliverable description:** This deliverable consists of a preliminary image search prototype based on textual entities. This is the basis for D 6.5, which will extend the textual component by NLP and multi-linguality.

**Due date of deliverable:** M18

**Actual date of deliverable:** June 30th, 2018

**Authors:** Duncan Blythe

**Project partners:** Zalando, Fashwell

**Workpackage:** 2

**Dissemination Level:** **public**

## Change Log

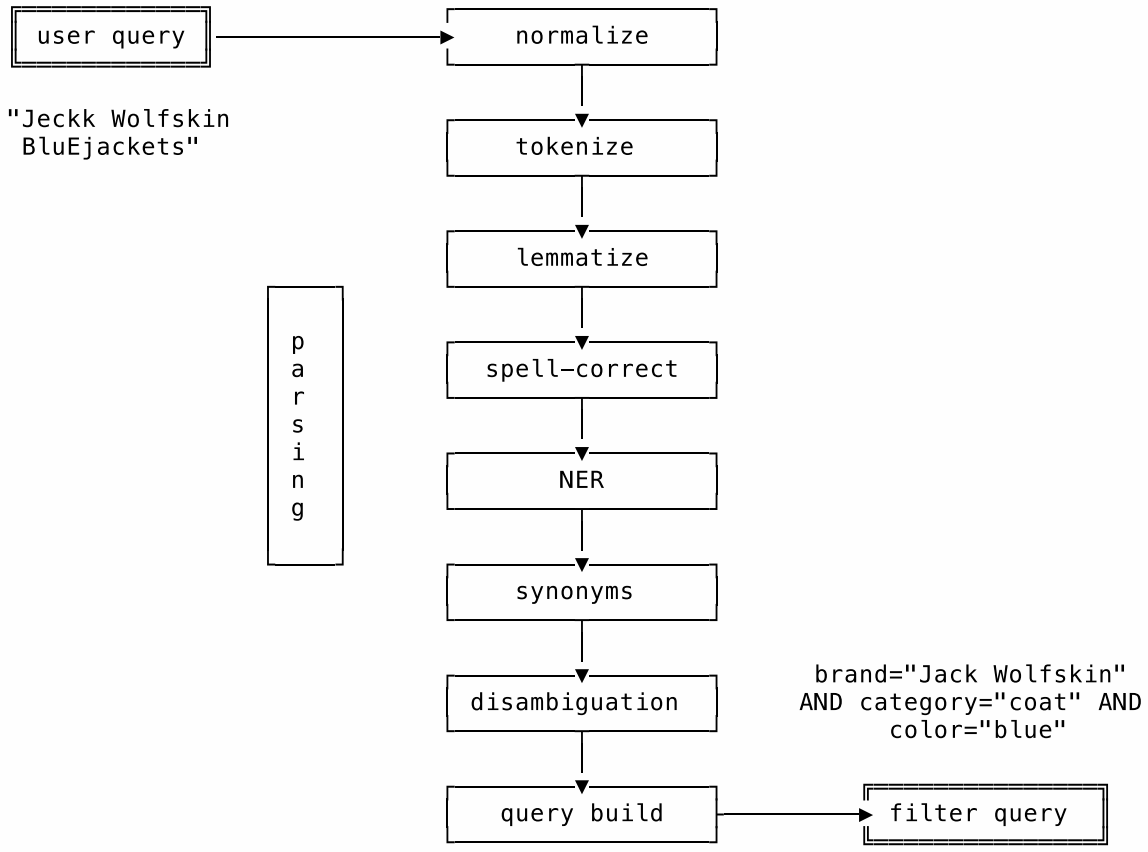
| Version | Date | Status | Author (Partner) | Description/Approval Level |
|---|---|---|---|---|
| 0.1 | 30.06.2018 | | Zalando | Confidential |

# Demo

This deliverable consists of a preliminary image search prototype based on textual entities. This is the basis for D6.5, which will extend the textual component by NLP and multi-linguality.

## Introduction

The product search system at Zalando currently uses a cascaded architecture to perform full-text search of the product data base. The products are indexed by attributes which accompany the products, either added in the course of the manufacturing process or enriched in the product curation process. The search architecture follows a cascaded design with input strings preprocessed as displayed in the figure below:




In the process of this cascade, several key nlp tasks are implemented which lead to a structured query which can be executed on the product data base. Although this approach is well-tested and understood, it suffers several disadvantages. These include but are not limited to:

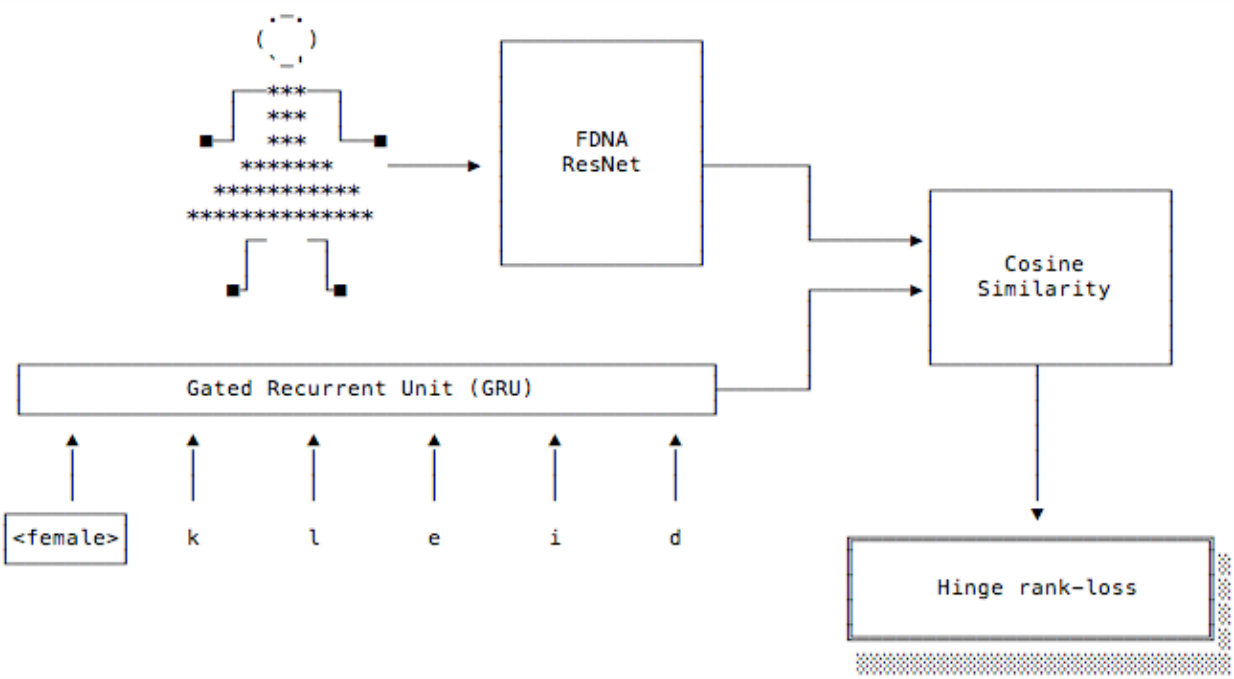
- Unclear relationship between component parts and end goal of improving retrieval
- Subtasks are not strictly necessary for retrieval
- Does not easily scale to new languages without language experts and additional dev ops
- Architecture fragile and may be broken by bugs in any of component parts

- Sequence of operations is arbitrary with many recurrent dependencies
- Retrieval is restricted to attributes explicitly represented in product data base
- No image-level information encoded

As a first step towards solving these issues this deliverable describes an alternative system which optimizes the mapping between query string and a product representation based on recent advances in deep learning. Rather than applying a cascade of operations we propose mapping the query string and products directly to a vector space in which matching products and queries are "close" and otherwise "separated".

# Architecture

We adopt an architecture loosely based on DSSM (Huang et al. 2013) and trained with a rank-loss hinge objective. An overview of the architecture is seen in the image below:




We represent the query string (exemplified in the image by "kleid") as a sequence of one-hot vectors in a small vector space, with dimension equal to the number of characters in the alphabet we are interested in (in this case the alphabet over queries in the Germany app-domain).

We add the gender category in which the user is searching to the beginning of this string. The string is then passed one-character at a time to a type of recurrent neural network, viz. gated recurrent unit (GRU) neural network (Chung et al. 2014) and the final hidden state is extracted.

More exactly, our sequence of one hot vectors is $x_0, x_1, \ldots, x_T$. Each one hot vector is then embedded in a vector space to give $u_t$ and updated linearly using the non-linear recurrence of the GRU ($f$).

$$u_t = W_x x_t$$
$$h_t = f(u_t, h_{t-1})$$

This final hidden state then represents the user-textual input.

On the image side, we encode all images in the current product data using a deep-residual convolution neural network (resnet) (He et al. 2016). So for a product image $I$, the network maps to a vector $g(I)$. Given $h_t$ and $g(I)$ we may compare the vectors and formulate an objective function which forces images and text which match to be "close" and otherwise "far" from one-another. To measure close-ness we use cosine-similarity:

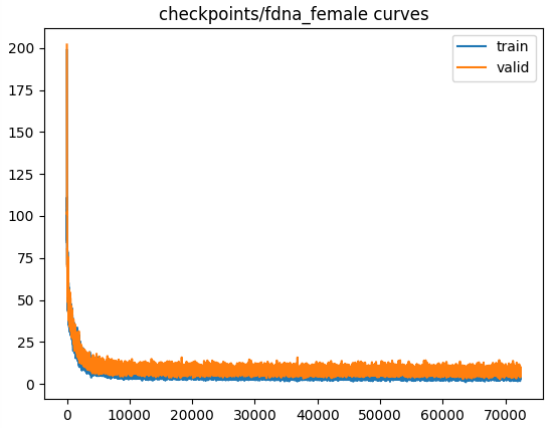
$$\text{css}(x, y) = \frac{x^\top y}{\|x\| \, \|y\|}$$

Our rank objective (Trotman 2005 enforces matching by encouraging matching image and text to be similar to another up to a threshold $\lambda$. Let $I^+$ be a match to $h_T$ and $I^-$ a mis-matching image. Then:

$$\mathcal{L}(h_T, I^+, I^-) = \max(0, \lambda + \text{css}(h_t, g(I^+)) - \text{css}(h_t, g(I^-)))$$

# Model Training

We perform stochastic gradient descent using the above loss function averaged over mini-batches, gradually reducing the learning rate during training. The development of the loss on the training and validation folds are displayed below:




# Quantitative Evaluation

In our experimentation we vary the hyper-parameter $\lambda$, the minimum number of examples required for a query to appear in training ("n counts") and whether we additionally include an embedding of attributes with the deep-residual features ("attr").

| n counts | use fdna | use attr | n hidden | lambda | top 1 | top 10 | top 100 | top 1000 | percentile |
|---|---|---|---|---|---|---|---|---|---|
| 1 | True | False | 1024 | 0.2 | 9.10 ± 28.76 | 41.40 ± 49.25 | 79.30 ± 40.52 | 95.50 ± 20.73 | 1.51 ± 6.07 |
| 5 | True | False | 1024 | 0.2 | 10.20 ± 30.26 | 41.30 ± 49.24 | 79.80 ± 40.15 | 95.60 ± 20.51 | 1.47 ± 5.70 |
| 20 | True | False | 1024 | 0.2 | 7.90 ± 26.97 | 36.10 ± 48.03 | 74.10 ± 43.81 | 93.80 ± 24.12 | 2.12 ± 7.56 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | True | True | 1024 | 0.2 | 8.70 ± 28.18 | 34.80 ± 47.63 | 70.10 ± 45.78 | 93.30 ± 25.00 | 1.87 ± 5.36 |
| 5 | True | False | 1024 | 0.1 | 4.50 ± 20.73 | 25.60 ± 43.64 | 62.30 ± 48.46 | 91.10 ± 28.47 | 2.82 ± 7.59 |
| 5 | True | False | 1024 | 0.3 | 7.70 ± 26.66 | 35.30 ± 47.79 | 76.20 ± 42.59 | 94.60 ± 22.60 | 1.79 ± 6.45 |

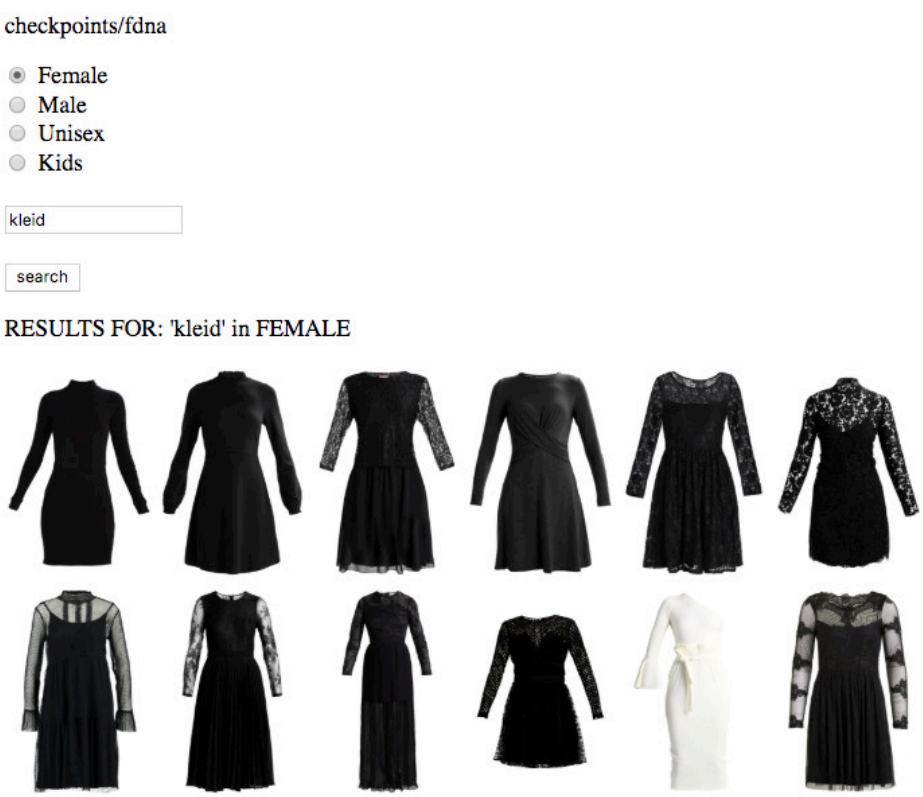
We find that the model using 5 counts, only residual featues ("fdna=True") and a hyper-parameter $\lambda = 0.2$ to perform best.

# Qualitative Evaluation

A good fashion product search system should fulfil the following desiderata:

1. Retrieving items based on basic attributes such as color, silhouette, brand
2. Meaningfully use gender as a basis for restricting the search domain
3. Be robust to spelling errors
4. Be able to work with abstract concepts over and above simple attribute types
5. Suggest meaningful products even when a requested item is not present in the data base

Retrieval based on silhouette:




Retrieval based on silhouette and color:

checkpoints/fdna

- (•) Female
- ( ) Male
- ( ) Unisex
- ( ) Kids

rotes kleid

search

RESULTS FOR: 'rotes kleid' in FEMALE

Retrieval based on silhouette and color-concept:

checkpoints/fdna

- (•) Female
- ( ) Male
- ( ) Unisex
- ( ) Kids

buntes kleid

search

RESULTS FOR: 'buntes kleid' in FEMALE

Retrieval based on brand leading to multiple types of product in brand:

checkpoints/fdna

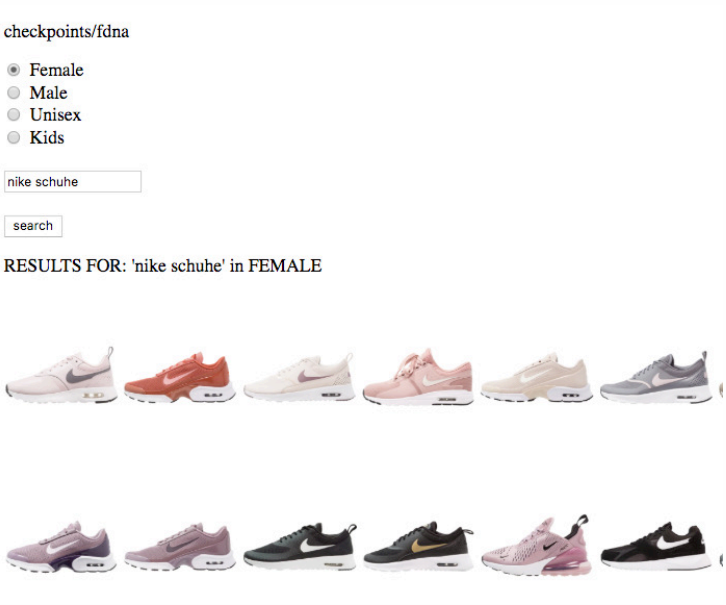
- (•) Female
- ( ) Male
- ( ) Unisex
- ( ) Kids

adidas

search

RESULTS FOR: 'adidas' in FEMALE

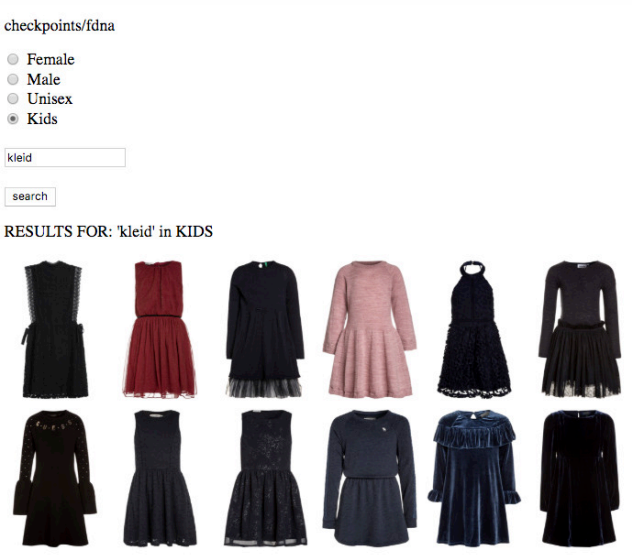
Retrieval based on brand and silhouette:



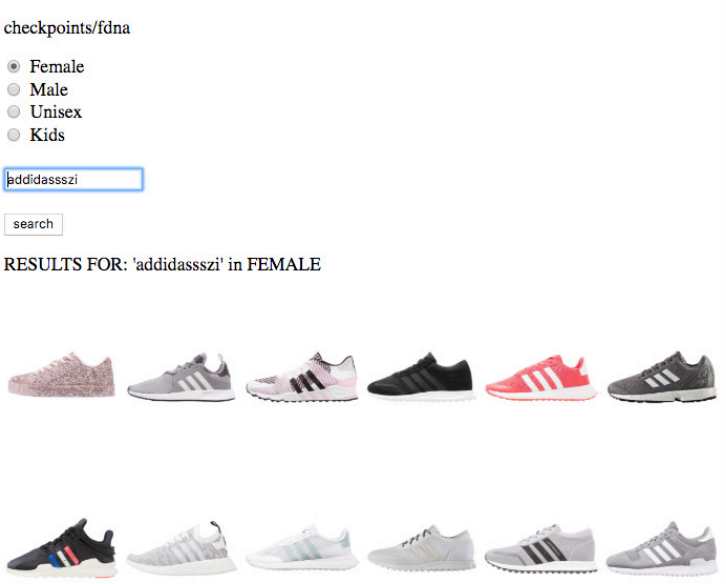

Gender selection based on visual attributes:


Male jeans tend to be baggier:

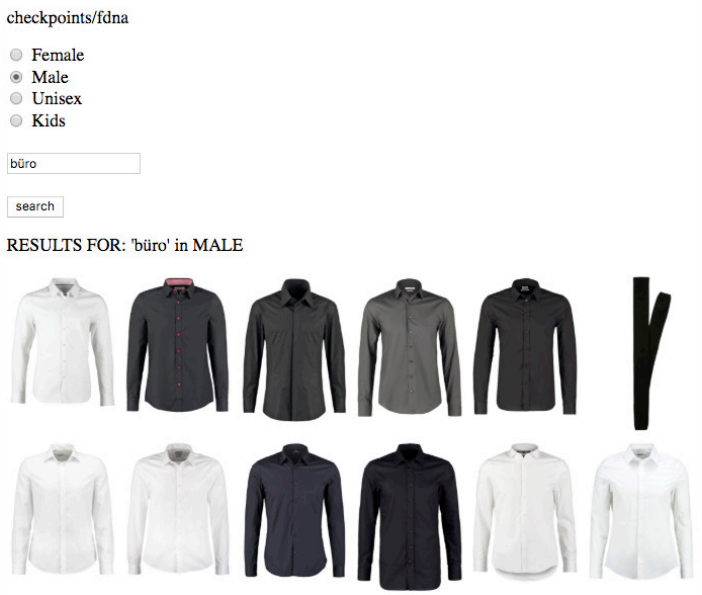
Also works in the children category:




The search engine is very robust to spelling:




Abstract concepts are also learned:



The football category also recommends related products:


# References

Huang, P. S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013, October). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (pp. 2333-2338). ACM. (url)

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. (url)

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). (url)

Trotman, A. (2005). Learning to rank. *Information Retrieval*, *8*(3), 359-381. (url)