ARCADA

Automized data collection process for a personalized online shopping platform

Hilde-Marie Fredriksson

Degree Thesis

Information Technology, Big Data

2019

Hilde-Marie Fredriksson

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik, Informationsanalys
Identifikationsnummer:	17490
Författare:	Hilde Fredriksson
Arbetets namn:	Automized data collection process for a personalized online shopping platform
Handledare (Arcada):	Jonny Karlsson
Uppdragsgivare:	

Sammandrag:
Detta slutarbete beskriver ett utvecklingsarbete gjord för en startup som utvecklar en personifierad ecommerce-plattform. Plattformen hämtar produkter från olika försäljare och därför är samlingen av data en viktig del. För att samlingen skall vara effektiv och snabb, måste gamla, manuella lösningen ändras. Slutarbetet beskriver utvecklingsarbetet från en helt manuell process till en semi-automatiserad process. Kraven för detta praktiska arbete, planeringen, utvecklingen och resultaten diskuteras i arbetet. Metoderna som använts är att hämta produkterna via en API, spara ID:n i databasen och sedan hämta de filtrerade produkterna från APIn. En översikt om existerande forskningar presenteras i kapitel 2 och efter det går arbetet igenom den praktiska delen av arbetet och sammanfattningen.

Nyckelord:	API, datainsamling, taggning, automation, bildigenkänning
Sidantal:	29
Språk:	Engelska
Datum för godkännande:	16.12.2019

DEGREE THESIS	
Arcada	
Degree Programme:	Information technology, Big Data
Identification number:	17490
Author:	Hilde Fredriksson
Title:	Automized data collection process for a personalized online shopping platform
Supervisor (Arcada):	Jonny Karlsson
Commissioned by:	

Abstract:
This thesis describes a practical work done for a startup that is building a personalized online shopping platform. The platform fetches products from different retailers and thus the data collection is a crucial part. For it to be effective and prompt, it had to be changed from the original, manual process. The thesis describes how the manual data collection process looked before this practical work and how it was changed to a semi-automated one. The requirements for such change, planning, development and results are discussed in this thesis. The methods used were fetching the products via an API, saving the IDs to the database and fetching filtered products for the user via the API. An overview of the existing research in the field of automatic attribution is given in chapter 2 and after that the thesis goes through the practical work and the conclusion.

Keywords:	API, data collection, tagging, automation image recognition
Number of pages:	29
Language:	English
Date of acceptance:	16.12.2019

CONTENTS

Figures

1 INTRODUCTION

1.1 Background

Personalization is the trend today and companies strive for making more and more personalized content. The e-commerce and fashion industry are no exceptions. The fashion industry is growing in revenue every year and purchases online increase (O´Connell, 2019). A study conducted by Entrepreneur magazine shows, that 51% of Americans prefer to shop online. However, according to an article published in Fashionunited (Van Elven, 2018), referring to a study conducted by Oliver Wymann, 75% of clothing purchases are still made in traditional physical stores. This can be reflected in the fact that user experience is underdeveloped compared to a customer visiting a physical store. People have increasingly busy lives, no time to spend online and want to shop efficiently. Without personalization, people end up scrolling for hours through irrelevant clothing and maybe leave the shop without buying anything at the end. This is not convenient for either the retailer or customer. Automation and personalization are important tasks to consider in the coming years, when inventory only grows, and the number of shoppers increase. At the same time, the crowd is more demanding and determined.

The background of this thesis is a matter of a startup, that is solving the problem of personalization in fashion e-commerce. The idea is to recognize each customer's style, gather products from several e-commerce stores and filter the product feed to match only her preferences. By doing this, the retailer gets highly targeted marketing and the customer does not have to be frustrated anymore. Instead she can just scroll through products that she could already see in her closet. This will increase the conversion rate greatly.

The problem that is discussed in this thesis is changing a manual process of bringing products to a platform to a semi-automated one. For the reasons stated above, tagged product data is needed and without image recognition this is quite manual work.

The startup in question is a service, that has developed an algorithm that provides recommendations of products on basis of the customer's style. The style is mapped out with a short questionnaire on the user end and the algorithm maps out corresponding styles on basis of the user's answers on the back end. This means, that they do not have any inventory, shipping costs or any of the basic retailer struggles. It's fully a service, which fetches its data from other retailers and adds needed parameters on its end to be able to recommend products.

Prior to this thesis, the problem was solved as follows: An affiliate deal with retailers was made, which gave access to their product feed. Product data was saved manually from a CSV-file (Comma Separated Values) to a JSON-file (JavaScript Object Notation), where in addition to basic product data such as price and name, the products were tagged with certain parameters. These parameters allowed the algorithm to fetch only products that matched the user's style. This method was incredibly time-consuming and not profitable at all to any parties. It led to the fact, that products were not added daily or even weekly, which was expected from the customers as well as the startup. However, for the time being this structure was serving the needs. This very primitive way of storing data was chosen because of test purposes - knowing, that it had to be changed in the future.

The practical work done for this thesis has included changing the way of storing product data, where it is fetched from and minimizing API-calls (Application Program Interface) in the process. Since an e-commerce platform is serving users with many different styles, there also must be a significant number of products to shop from. The usage of APIs makes the gathering of data much more efficient and less time-consuming. In addition to this, the product feed showed to the user will be real-time, with prices and stock inventory updating whenever the page is refreshed. In this practical work only one API was used, the ASOS API from RapidApi.

1.2 Purpose

The purpose of the practical part of this thesis has been to develop a semi-automated process of saving product data, tagging it and displaying personalized product feeds to the users. This thesis will describe the practical work, motivate the decisions and analyze the success rate of the change. The thesis will not discuss the tagging process nor the algorithm in detail. The references to these will be vague, to protect startup secrets.

1.3 Methods

The choice of tools for this development work was based on the author's experience and research. The programming language in question is Angular 8 with Typescript. The technical stack includes Firebase authentication, Firestore database, RapidApi API for ASOS, and various libraries in Angular 8. APIs were chosen as the method of fetching product data instead of the other possibility, web scraping. This choice that ruled out web scraping was made because of large amount of data, saving database space and deficiency of real-time data.

1.4 Structure

The rest of the thesis is structured as follows: Chapter 2 will discuss how tagging of clothing and personalization is done today in ecommerce stores. In chapter 3 an overview of the solution prior to this thesis is given. The change to a semi-automated solution and the development work is described in detail in chapter 4. Discussion of the results will be done in Chapter 5.

2 TAGGING SYSTEM OF CLOTHING APPAREL

When a customer walks into Macy's and asks for a "romantic, V-neck blouse in red", an employee will instantly walk up to her with a matching product and the customer walks out in minutes with a piece of clothing that matches her style. A customer trying to experience the same at an online store – the result will be utterly disappointing, and in most cases the customer will end up spending hours browsing, maybe ending up buying nothing. Taking the fine-grained attribution to the next level and being able to categorize what the overall style is, would enhance the personalization at fashion online stores greatly. As personalization is becoming the most important factor in the online world (Maff, 2019), one could image that a 110-billion-dollar industry (Clement, 2019) like online fashion would have solved this by now.

All ecommerce sites do some kind of personalization, but many of them learn from the user's actions, follow clicked products or keywords that were searched for. A 'cold start'-problem applies to the online retailers (Iliukovich-Strakovskaia, 2018). Cold start means, that personalization won't happen until the user interacts with the platform e.g. clicks on products, favorites them etc. There are many layers to the problem, first one being the automatic tagging so that the products can be filtered without human interaction. Secondly, they should be personalized to the user's preferences but commonly this happens only after the user has interacted with the platform.

The core idea of the startup in question is based on research in fine-grained tagging of apparel and how to map those out to recognize a person's style. The point is to turn the basic personalization around, and have the user give their likes and dislikes upfront and then the model would learn from the user's behavior to enhance the recommendation even more. This way, the user gets a 100% personalized content from the first second on the

platform. In this chapter I will go through what kind of solutions exist, how they work and where they deficit.

2.1 Research in tagging systems

A lot of research has been done in this field to automate the tagging process of apparel field (Khosla, 2015). Working examples can be found in the area of tagging clothing categories and are being used in the field of ecommerce – e.g. if it is a blazer or a skirt. Research for clothes tagging includes a study by Kalantidis et al. (2013) that suggests a cross-scenario retrieval of clothing. However, for a machine to learn subtle nuances of for example whether it is a turtleneck, or a high neck can be very challenging. This is a quite an active research topic, since automizing fine-grained categorizing is a needed feature in the future of the growing ecommerce.

Apparel Classification with Style (Bossard et al., 2012) introduces a pipeline for recognition and classifying not only apparel type but also style features such as patterns, color, material etc. Research made by Liu et al. provides a great overview on existing solutions and suggests a similar clothes retrieval based on fine-grained attributes as well as a dataset called DeepFashion (Liu et al., 2016). Di et al. (2013) presents another dataset called WFC which includes jackets and coats and fine-grained clothing style recognition with supervised learning features.

Various techniques have been used for the image recognition, either supervised or non-supervised learning. According to the studies mentioned above, the supervised learning uses hand crafted features such as SIFT or HOG. These features come with good capability of tackling occlusion which deep models have a hard time with. However, these have limited performance and thus deep learning methods have been taken into action. Deep models have been introduced to learn more discriminative representation in order to handle cross-scenario variations.

Xiao et al. (2015) introduces a general framework to train convolutional neural networks with noisy labeled data. These methods achieve good performance (Xiao et al. achieved a 78.24% accuracy), but they ignore the deformations and occlusions in the images, which

hinder further improvement of the recognition accuracy (Liu et al., 2016). Iliukovich-Strakovskaia et al. (2016) suggests a solution based on pre-trained deep neural network models, which reached a 69.3% accuracy. This model is suitable for small collections of images because it allows obtaining qualitative feature representation enriched with knowledge from external crafted datasets.

State-of-the-art research includes Hou et al. (2019) who propose a Semantic Attribute Explainable Recommender System (SAERS) with fine-grained labeled data and pre-trained a Semantic Extraction Network with CNN (Convolutional Neural Network). Zakizadeh et al. (2018) tailored the DeepFashion dataset for training fine-grained attribute recognition models. DeepFashion is a dataset introduced by Liu et al. (2016) and has been widely used for automatic attribution purposes. Deng et al. (2018) have come very close to a functional personalized clothing recommendation algorithm, which includes automatic tagging and recommendation of clothing according to the attributes and style of the products. They used a mix of classical feature extraction and deep learning, which seems to be a good recipe for future development.

2.2 Commercial tagging methods

There are ecommerce stores that use image recognition models to recognize e.g. the neckline – whether it is a V-neck or an O-neck and thus can easily categorize the item to fit one filter. This is a rather new development and has been taken into action by e.g. Urban Outfitters. They use Google's AutoML Vision, which also brings a new layer to the technologies by being a model that a non-technical person can train (Li and Li, 2018). There are many possibilities in the field, and a lot of people try to crack this difficult task.

Many big ecommerce stores have resources to develop their own models to help with product tagging and visual search. Zalando for example has trained a model with deep learning for image retrieval based on eight apparel types. Zalando has also created a freely available dataset called Fashion MNIST for machine learning challenges like these (Lasserre, 2018). ASOS has used a hybrid-model to tackle the 'cold start'-problem, by model composition and simultaneous optimization. ASOS has a lot of manually annotated product attributes which are missing. They developed a text-classifier with CNNs to tag

the products according to their product descriptions, which were sentences (Cardoso, 2018).

Not many look at this from a personalization perspective, but rather automizing inventory and category filtering. The startup in question, however, is considering personalizing as their priority since their whole core idea is based on that.

2.3 Personalization with the help of tagging

The earlier mentioned factors in mind, the startup started developing their idea, which would be automized in the future. Even if the tagging of fine-grained features was automized, there still would not be a lot of improvement to the personalization – they could only filter out certain attributes that the user does not like. It is, however, also an important feature. For the startup's idea though, it was important to map out the 1000 most common attributes used in apparel. As a reference to the attributes, the dataset in the paper by Liu et al. (2016) was used. With these they could map out which attributes made which style and that is the feature that brings the unique value to the startup. However, this is a challenging task since stylistic descriptions vary by geographic location and season – just to name a few. In the end the model would learn from the user's actions on the platform and give even more accurate suggestions. This is where the actual personalization is done and where the algorithm is used.

This thesis explains how this is done manually at the time and the practical work of this thesis is changing it to a semi-automated solution. The final solution and future development will be fully automized with machine learning, however, this is out of the scope of this thesis.

3 THE ORIGINAL PROCESS FOR MANUAL DATA COLLECTION

3.1 Description of the old solution

The original process for manual data collection was lacking in efficiency. The main actions needed for getting all the needed product data for the algorithm to work, were the following:

1. Download a CSV-file from the affiliate network with the retailer's product feed.
2. Look for products from the file, copy parameters such as: product name, price, the URL to the affiliate link and image URL (figure 1).
3. Save thumbnail picture on server
4. Add three additional parameters to the product, which were set by the startup in question and was not possible to get from the retailer
5. One product was done.

This process took around 5 minutes. As the startup needed several hundred products – if not thousands and in addition to that tens of products should be added every day, it's clear that this solution was far from profitable.

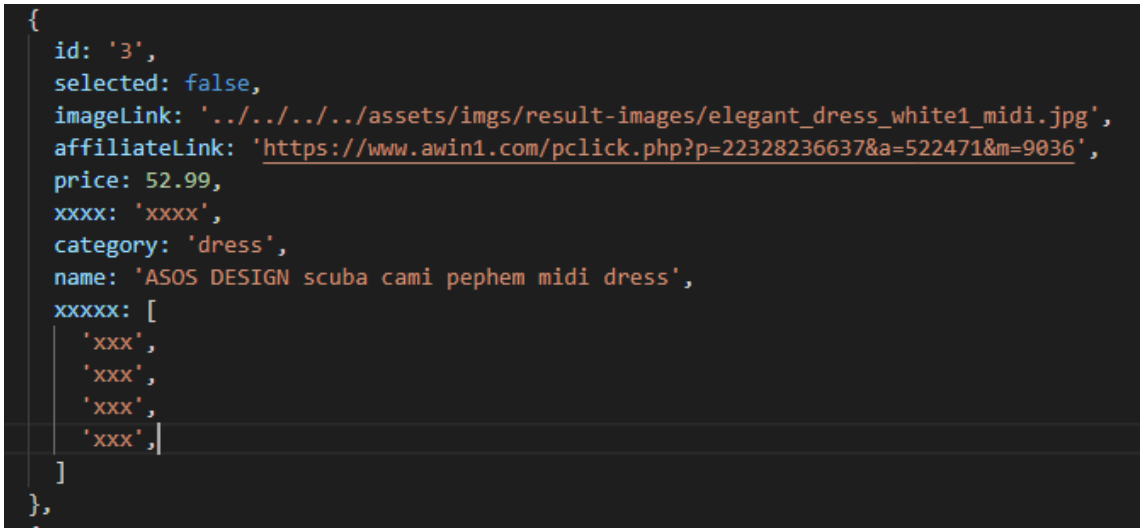

Figure 1. One product in the JSON-file

This data was static, which led to disappointments when the user found a product that she wanted to buy. When she clicked on the product and was taken to e.g. asos.com, the product could be out of stock. Products are added and removed frequently in online stores, thus the selection changes quite often. It's already a long funnel to take the customer through to buy something. This possible disappointment decreases the conversion rate even more.

As explained in the introduction, this primitive way was chosen mainly because of the startup being in demo-phase. There was a lot of testing going on which was focused on the algorithm, rather than focusing on the efficiency of adding products. So, for the time being the solution was acceptable.

However, the main testing phase is over for the startup. A startup is very agile and lean, since there are no bureaucratic processes to go through when change is needed, but sometimes it comes with more workload in the end.

3.2 Pros and cons of changing to an automatic process

As stated earlier, the old process was very inefficient, time-consuming and the result was static data. When changing the process to API- and database based, the data will be in real time. Every time the user comes to her personal shop, the page will make a request to the product data. This means, that if a product's price changes, the user will see this instantly in her personal shop. This way no products that are out of stock will be displayed either.

The Firebase Firestore database stores only the product ID, URL and the parameters added by the startup for the algorithm filtering process. Rest of the data comes from the API, which is very light weight. Since the parameters are mandatory for the recommendation to work, there must be some manual work included in the process. If the technology was developed enough, this could be automated with image recognition. However, a model taught for that does not exist at the time of writing.

The risk that comes with this solution, is that the whole base of the startup relies on the API. If it is taken down, then the business must, very quickly change to another API or change back to static data. Also, it can become costly when the page gets a lot of traffic, since every request costs. Unfortunately, these are the risks that must be taken in order to get real time data.

The other way considered was web scraping. However, this was ruled out because of large amount of data, saving database space and the deficiency of real-time data. Web scraping is something that will be considered in the future, outside of this thesis, when there's a need for more products outside the stores that provide an API. The end solution would be a mix of real time and static data, which is the most common way of platforms like the one in question.

4 PLANNING AND DEVELOPMENT OF A SEMI-AUTOMATED DATA COLLECTION PROCESS

4.1 Requirement specification

The requirements for the semi-automated data collection process were decided on basis of these questions: What were the main points that needed to be changed, what was already good and what actions had to be taken for the change to be fulfilled?

When these were considered, the requirements were specified as follows:
1. The need of different APIs, preferably multiple to get as many products as possible
2. The attribution process is done by non-technical people, so a clear database structure and parameter naming is mandatory
3. Integration to the old data should not affect the already registered users
4. As much as possible data should come from the API
5. A timed batch needs to exist to bring new products for tagging automatically
6. The products need to have a time stamp, so that old products can be removed and the product page URL which will be replaced with an affiliate link

4.2 Planning

The planning of the change to a semi-automated process began with mapping out the requirements, which were already described in the previous section. The possibility of using APIs was dependent on the APIs available at the time. Retailers are not exactly fond of sharing their product data with others, so there are not many available. However, ASOS and H&M have a public APIs that you can access with an API-key. It allows you to get all product data. This was the first and foremost thing to consider. Without this, the change would have not been possible. For this practical work, only ASOS API is considered.

For being able to structure the data correctly, a good look had to be taken on what the API returns and in what format. When this had been gone through, the planning of structuring the database could begin. Here it was important to remember, that the integration should

go through without any issues and that the startup's non-technical people could easily tag the products. There are already a thousand users on this platform, and the change of this should not affect them in any way. The data had to be structured in the same way as earlier. It was decided that the database of products would only contain the product ID, URL of the product page, and parameters added by the startup, as well as a timestamp. The product ID is fetched from the API which matches the product ID coming from the database (figure 2). All the rest of the data displayed to the user comes from the API (figure 3). The added parameters in the database are for the algorithm to filter out the correct products to be displayed.

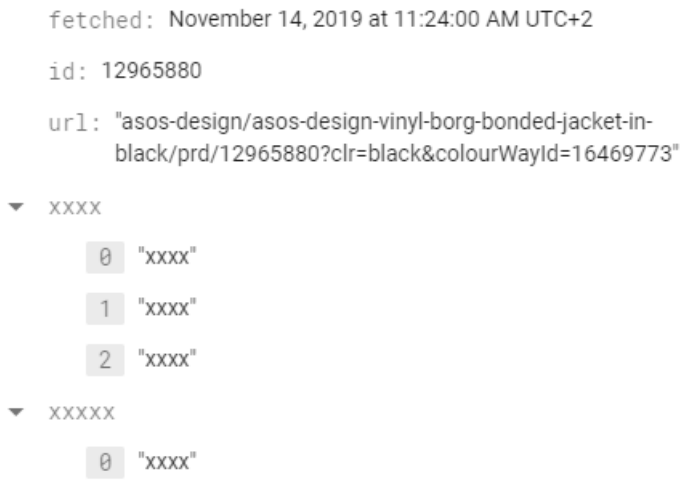

Figure 2. Database structure, one product

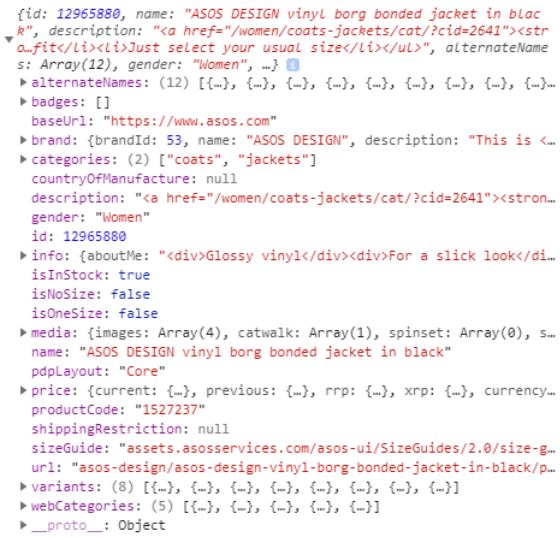

Figure 3. Structure of the API-response of one product

A timed batch run runs e.g. once a day or once a week and saves new product IDs and URLs to the database. The one responsible for product tagging then goes through the products and adds the needed parameters. Then the products are ready to be displayed for the user. This shortens time per new product massively and hence the startup profits when they can add new products more often with using less time.

The API in question is not built in the best way, unfortunately. A well-built API for the needed purpose would allow the API call to be done with multiple product IDs. The one used in this practical work allows only one ID per call. This is not cost-efficient by any

standards, but manageable. It means that when programming, some actions towards decreasing the amount of API-calls had to be taken.

In addition to these, there were also some mapping issues to consider with the data received from the API. In the ASOS system all product sizes, categories etc. are numbered. For example, instead of getting ‘category: jumpers’, the response is ‘category: 3405’. For the startup’s sake and since ‘category’ is a value that’s displayed to the user, these values had to be mapped to actual words.

4.3 Development

The biggest workload was updating the old code and integrating the algorithm to match the new solution. Since there were many parameters to consider in recommending products, it was a step-by-step solution in the code.

When the user profile from the algorithm was matched with the product database, a lot of filtering had to be done to get the correct products that match the style profile of the user. This included arrays that were filtered and pushed into new arrays, each time when data had to be filtered according to some requirement (figure 4). When all the suitable products were collected, they had to be divided into chunks of 20 products each to reduce the API-calls.

The API-call was requested with one id from the chunk (figure 4). At the same time the categories that the product belonged to were checked and mapped to words (e.g. shirt instead of 2349). Also, before pushing the product to the final array, there was a check for whether the product was in stock or not.

```
this.http.get
  ('https://asos2.p.rapidapi.com/products/v3/detail?currency=EUR&store=ROE&lang=EN-GB&id='
  + id
  , requestOptions)
  .pipe(map((response: any) => response))
  .subscribe(data => {
    this.product = data;
    this.product.url = url;
    this.product.categories = [];

      for (let i = 0; i < this.product.webCategories.length; i++) { // loop through product categories
        for (const [key, value] of stringCategory) { // look through all categories
          if (key === this.product.webCategories[i].id) { // if key of all categories == product category
            this.product.categories.push(value);
          }
        }
    }
      if (this.product.isInStock !== false) {
        this.finalApiProductsArray.push(this.product);
      }
      this.displayHowMany = this.finalApiProductsArray.length;
});
}
```

Figure 4. API-call to get one product. The final view will only contain products that are in stock.

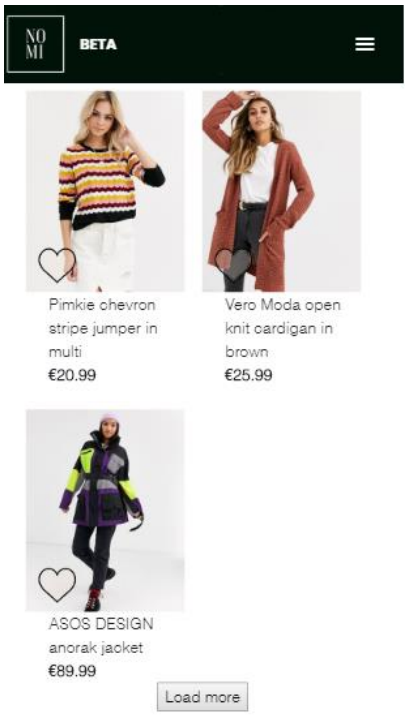

Figure 5. End view with products filtered down to suit the user's style and brought from the API

To decrease the number of API-calls, the following actions were taken: When the user arrives to her personal store, a function calls 20 products at once. These, because of reasons stated earlier, are 20 separate requests. If the user wants to see more products, she presses 'load more' button and 20 more requests are made (figure 5). This way only the needed API-calls are made, the page is not overloaded, and the performance stays good.

The new system was not compatible with the earlier 'favorite' action on the platform. In the old version this had been done with form controls, but now that the data was not static anymore, this had to be changed in the code. The code behind displaying of favorited products had to be changed as well, to the same logic as the product feed.

Some of the old version's functions could still be used, but many required some modifications. Furthermore, the development work included some UI-changes, when a product page was added (figure 6). This made the platform more like an actual online store and more shoppable.

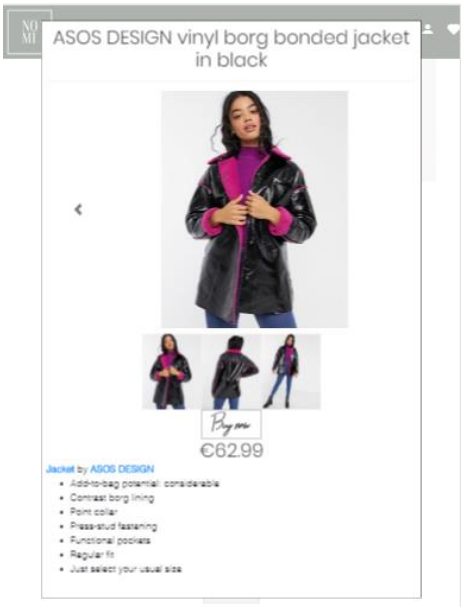

Figure 6. Product page with all data coming from the API in real time

4.4 Testing

When testing the semi-automated process, the code for fetching product IDs was run and the one responsible for tagging the products went through 20 products. The tagging process was timed, and the new solution timed 1.5 minutes per product. That is a big decrease from 5 minutes. This equals to 20 products added in 30 minutes vs. the former 100 minutes. The most time taken is not the actual tagging but figuring out which parameters the item matches with. This is something that will be automated in the future with image recognition.

Testing included faulty data in the database, and what happens when that occurs. If faulty data is found in the database, the code will ignore it and not display it to the user. Only when a matching product has been found from the API, the product is displayed. Also, if the API response returns error, the solution is the same – the product is not displayed at all.

4.5 Results

The requirements of the semi-automated data collection process were fulfilled well, and the practical part of this thesis will be taken to production as soon as the new design is finished. The amount of time to get one new product to the platform decreased dramatically (the old time was 330% longer). The time used per product was the most significant factor in this practical work, and that criteria was fulfilled better than well. In the future, the startup will develop an image recognition model that will do the tagging automatically. Then there's no manual work left at all and the business can scale easily.

The development work wasn't that complex but would have gone significantly faster if I would have more experience in the area. Angular with Typescript is not unfamiliar for me, but I learned more again during this practical work. I am more confident to continue development. This was the first time I did anything with APIs, so I am glad that it went this well.

4.6 Future work

Future development will include adding several APIs instead of just one and considering web scraping to get even more products on the platform. After this, a model for recognizing the needed parameters automatically will be developed, most likely with deep learning methods which have evolved rapidly in the past years. The future work includes automatic attribution of fine-grained clothing attributes such as ruffle or bow as well as looking at the overall style of the product in question. This will require a huge dataset with quality data of catalog pictures taken from various retailers.

For additional features such as shop by picture, in other words visual search there are many available options on the market, and they may consider using them. No product on the market at the time has good enough image recognition to be used for the fine-grained attribution. However, if a model suitable for this usage is developed, the startup could benefit from collaboration. The crucial part is the deep learning model for the style recognition which is the unique standpoint of the startup. It needs to be developed by the startup

itself and this will conclude a fully automated system, that will not need human interaction. That is the end goal of the startup.

5 CONCLUSION

In this thesis a practical work of changing a manual data collection process to a semi-automated one was documented. The background to this thesis is a matter of a startup. They have developed an algorithm that recognizes the user's clothing style and gives accurate recommendations on their platform according to the style profile. The manual data process was very primitive, slow and chosen for testing purposes - seeing that it had to be changed in the future. It included saving data manually to a JSON-file, which took around 5 minutes per product. The goal of the practical work done for this thesis was to make the data collection process prompt and efficient.

The change to the semi-automated process included fetching data from an API, saving product IDs to the database and fetching correct products according to the style profile of the user from the API. This semi-automated process led to a 330% faster data collection process compared to the manual process. With this the startup can add products to its database efficiently. The change was needed and is ready to be put into production as soon as the UI design changes have been done. Results were as expected, and still leave space for future development, which was a known fact. The big decrease in time per added product shows that there is absolutely no point in collecting manually, if the data is large.

The future development will include adding more APIs and web scraping as ways of data collection. Automating the tagging process fully with image recognition is the end goal of the startup. This means that every product goes through image recognition model using deep learning to recognize fine-grained attributes and recognizes the style of the clothing. Based on the existing research the best way to do this would be a mix of classical feature extraction and deep learning. No human interaction will be needed and thus it is easy to scale fast.

REFERENCES

Bossard L., Dantone M., Leistner C., Wengert C., Quack T., Van Gool L. (2013) Apparel Classification with Style. In: Lee K.M., Matsushita Y., Rehg J.M., Hu Z. (eds) *Computer Vision – ACCV 2012. ACCV 2012. Lecture Notes in Computer Science*, vol 7727 (pp. 321–335). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37447-0_25

Deng, Q., Wang, R., Gong, Z., Zheng, G. and Su, Z., 2018, November. Research and Implementation of Personalized Clothing Recommendation Algorithm. In *2018 7th International Conference on Digital Home (ICDH)* (pp. 219-223). IEEE. DOI:10.1109/ICDH.2018.00046

Hou, M., Wu, L., Chen, E., Li, Z., Zheng, V.W. and Liu, Q., 2019. Explainable Fashion Recommendation: A Semantic Attribute Region Guided Approach. *arXiv preprint arXiv:1905.12862*. Available at: https://arxiv.org/pdf/1905.12862.pdf [Accessed 21 Nov. 2019].

Iliukovich-Strakovskaia, A., Dral, A. and Dral, E., 2016. Using pre-trained models for fine-grained image classification in fashion field. In *Proceedings of the First International Workshop on Fashion and KDD, KDD* (pp. 31-40). Available at: http://kddfashion2016.mybluemix.net/kddfashion_finalSubmissions/Using%20Pre-Trained%20Models%20for%20Fine-Grained%20Image%20Classification%20in%20Fashion%20Field.pdf [Accessed 21 Nov. 2019].

Kalantidis, Y., Kennedy, L. and Li, L.J., 2013, April. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval* (pp. 105-112). ACM. Available at: http://image.ntua.gr/iva/files/kalantidis_icmr13.pdf [Accessed 18 Nov. 2019]

Lasserre, J. (2018) Shop the Look with Deep Learning. *Zalando technology blog*, Sep 12. Available at: https://jobs.zalando.com/tech/blog/shop-look-deep-learning/ [Accessed 18 Nov. 2019]

Li, F. and Li, J. (2018) Cloud AutoML: Making AI accessible to every business. *Google cloud*, Jan 17. Available at: https://www.blog.google/products/google-cloud/cloud-automl-making-ai-accessible-every-business/ [Accessed 21 Nov. 2019].

Liu, Z., Luo, P., Qiu, S., Wang, X. and Tang, X., 2016. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1096-1104). Available at: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Liu_DeepFashion_Powering_Robust_CVPR_2016_paper.pdf [Accessed 18 Nov. 2019].

Maff, A. (2019) What are the benefits of personalization in e-commerce? *Seller´s choice*, 12 September. Available at: https://blog.sellerschoice.digital/benefits-personalization-e-commerce [Accessed 21 Nov. 2019].

O´Connell, L. (2019). *Apparel: market growth rate worldwide 2020 | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/727541/apparel-market-growth-global/ [Accessed 18 Nov. 2019].

Van Elven, M. (2018) '75 percent of fashion purchases made at physical shops, but people spend more online'. *FashionUnited*, 8 June. Available at: https://fashionunited.com/news/retail/75-percent-of-fashion-purchases-made-at-physical-shops-but-people-spend-more-online/2018060821670 [Accessed 18 Nov. 2019].

Zakizadeh, R., Sasdelli, M., Qian, Y. and Vazquez, E., 2018. Improving the Annotation of DeepFashion Images for Fine-grained Attribute Recognition. *arXiv preprint arXiv:1807.11674*. Available at: https://arxiv.org/pdf/1807.11674.pdf [Accessed 21 Nov. 2019].

Xiao, T., Xia, T., Yang, Y., Huang, C. and Wang, X., 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2691-2699). Open access version available at: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Xiao_Learning_From_Massive_2015_CVPR_paper.pdf [Accessed 21 Nov. 2019].

Cardoso, Â., Daolio, F. and Vargas, S., 2018, July. Product characterisation towards personalisation: learning attributes from unstructured data to recommend fashion products. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 80-89). ACM. Available at: https://arxiv.org/pdf/1803.07679 [Accessed 24 Nov.2019]

Iliukovich-Strakovskaia, A., Tsvetkova, V., Dral, E. and Dral, A., 2018, March. Non-personalized Fashion Outfit Recommendations. In *World Conference on Information Systems and Technologies* (pp. 41-52). Springer, Cham. https://www.researchgate.net/profile/Anna_Iliukovich-Strakovskaia/publication/323962407_Non-personalized_Fashion_Outfit_Recommendations/links/5b325b820f7e9b0df5cc9d13/Non-personalized-Fashion-Outfit-Recommendations.pdf [Accessed 24 Nov.2019]

Khosla, N. and Venkataraman, V., 2015. Building image-based shoe search using convolutional neural networks. *CS231n course project reports*. Available at: http://cs231n.stanford.edu/reports/2015/pdfs/nealk_final_report.pdf [Accessed 21 Nov. 2019].

6 APPENDIX: SUMMARY IN SWEDISH

Detta examensarbete beskriver ett utvecklingsarbete gjord för en startup som utvecklar en personifierad näthandels-plattform. Plattformen hämtar produkter från olika försäljare och därför är samlingen av data en viktig del. För att samlingen skall vara effektiv och snabb, måste gamla, manuella lösningen ändras. Slutarbetet beskriver utvecklingsarbetet från en helt manuell process till en semi-automatiserad process.

Personifierande av innehåll är en trend i dagens värld och företag försöker hållas med i ändringen. Modeindustrin växer år för år och mängden inköp på nätet också. En studie gjord av Entrepreneur-tidningen visar, att 51% av amerikaner väljer att handla på nätet. Däremot görs 75% av klädinköpen fortfarande i fysiska butiker. Detta kan reflekteras till att användarupplevelsen är inte tillräckligt bra på nätet och det helt enkelt är lättare att hitta passande kläder i en fysisk butik. Folk har mer hektiska liv än någonsin förut och har inte tid att spendera på nätet. De vill hitta sina kläder snabbt och enkelt. Utan personifierande hamnar man bläddra igenom tusentals produkter och spendera flera timmar i nätbutiken. Kanske man till och med lämnar butiken utan att köpa något, eftersom man inte hittat ett passande klädesplagg eller man bara ger upp.

Metoderna som använts i detta praktiska arbete är följande: hämta produkterna via en API, spara ID:s i databasen och sedan hämta de filtrerade produkterna från APIn. Arbetet går också igenom existerande forskningar och hur de används kommersiellt. Beslutet att använda APIn istället för web scraping gjordes p.g.a. stor mängd av data, att spara plats i databasen och att data skulle vara statisk. Då data kommer via APIn är den realtid, och det enda som behövs sparas i databasen är produkt ID: n.

Alla nätbutiker gör någon slags personifiering, men största delen av dem följer användarens verksamhet på sidan, till exempel vad användaren klickar på eller vilka sökord hon använder. Företaget i frågan vill svänga om problemet, som uppstår då man personifierar med att följa användarens verksamhet. Då användaren anländer till majoriteten av nätbutikerna, görs ingen personifiering utan användaren ser en generell vy. Istället har företaget i frågan ett snabbt test, där användaren väljer vilka klädesplagg passar bäst i hennes stil samt preferenser om material, mönster och färger. Algoritmen går igenom svaren, och

bestämmer en stilprofil som passar åt användaren. Användaren får genast ett helt personifierat urval framför sig. Mindre och specifikt inriktat urval förbättrar konversionen.

Den originella processen var manuell och tidsdrivande. Det tog ca. 5 minuter att lägga till en produkt i databasen. Processen innehöll taggning av produkter, vilket består av att lägga till beskrivande attributer om klädesplagget. Också klädesplaggets stil måste läggas till manuellt. De här två parametrarna gör så att algoritmen rekommenderar produkter personligt. Eftersom företaget behöver hundra- om inte tusentals produkter i deras databas, är det klart att manuella processen inte var lönsam. Data var också statiskt, vilket ledde till besvikelser då användaren ville köpa produkten, och skickades framåt till originella försäljarens sida. Produkten kunde vara ’ej i lager’, eller rätt storlek fanns ej. Manuella processen valdes eftersom företaget var i testningsskede, och var mer fokuserad på att testa algoritmens funktionalitet istället för effektivitet av datainsamlingen.

Bytet till en API-baserad lösning skedde på basis av kravspecifikationen. Kraven var till exempel att så stor del som möjligt av data ska komma via APIn, integrationen till nya processen ska inte påverka de existerande användarna och taggningen ska vara möjlig för icke tekniska personer. Det finns inte många offentliga APIn för klädbutiker, men ett par existerar. För detta arbete användes ASOS API och via den kan företaget få tiotusentals produkter till sin plattform.

APIn är byggd så, att man kan kalla endast på en produkt i taget. Detta är inte väldigt lönsamt och effektivt, men man kan göra åtgärder för att få kostnaderna lägre. Man måste betala för varje API-begäran och eftersom en produkt betyder en API-begäran, kan inte alla produkterna laddas upp på en gång. Användaren ser 20 produkter på en gång, och kan sedan klicka på en ’ladda mer’-knapp, vilket gör 20 till begäran.

Största jobbet var att uppdatera gamla koden och att integrera algoritmen att fungera med lösningen. Då användarens stilprofil var anpassad till produktdatabasen, måste den gå igenom flera filtreringar så att användaren fick rätt produkter i sin vy. Ytterligare måste produktkategorierna som kommer via APIn konverteras till ord. Kategorin ’skjortor’ kunde till exempel vara ’2405’ och detta lämpade sig inte till företagets behov. Nya processen var inte heller förenlig med ’favorit’-funktionen och detta måste omskrivas i koden.

Då praktiska jobbet var avklarat, måste den såklart testas. Taggningsprocessen var tajmad och den gick ner till 1,5 minut från 5 minuter. Det här betyder att företaget nu kan lägga till 20 produkter i 30 minuter istället för det gamla 100 minuter. Testningen bestod också av felaktiga data i databasen, och vad som händer då man kallar på det. Koden är gjord så, att användaren inte ser några produkter om dom inte hittar en match från APIn.

I fortsättningen bör processen automatiseras totalt och detta görs med bildigenkänning. En modell måste utvecklas för att känna igen klädesplaggets stil samt att taggningen av attributen så som röd, bälte och a-linje görs automatiskt. Forskning inom området är aktivt och kan användas för utvecklingen av modellen.