Ultimate Capsule, an app that helps us dress better with less clothes

Susana Isabel Pires Machado Ferreira Guerreiro
Departamento de Engenharia Informática
IST-Taguspark, Universidade de Lisboa, 2744-016 Porto Salvo, Portugal
susana.f.guerreiro@tecnico.ulisboa.pt

ABSTRACT
The process of choosing an outfit is a considerable important task, because it influences the impression that we give to others. This is a complicated task due to higher number of clothes in people’s closets. The term capsule wardrobe was invented by Susie Faux, in 1980 and it has been reinvented in different fashion online blogs. Although it is a concept relatively famous, that we have knowledge of, there is no application or service capable of simultaneously allow users to create capsules wardrobe and obtain outfits suggestions based on them. Motivated by the capsule wardrobe concept, we developed an Android application, called Ultimate Capsule, that allows users to create capsules wardrobe and obtain outfits recommendations. Regarding outfit recommendation we added natural language to allow users to textually describe them. We followed an iterative and incremental development approach. Tests with users have shown that we were able to obtain an intuitive application, easy to learn and use.

Author Keywords
Clothes Coordination; Capsule wardrobe; outfit recommendation; natural language.

INTRODUCTION
According to [8], one of the visual aspects that influences the impression people have about us is the clothing items that we choose to wear and how we combine it. Having this in consideration, making the decision of choosing an outfit becomes a considerable important task task, however this is a very complex task due to the fact the number of clothing pieces in people’s closets has increased significantly.

In 1980, Susie Faux created the concept capsule wardrobe, which consists in the smallest possible set of clothing items, including shoes and accessories, all possible to combine with each other, producing several different outfits. Over the past few years, this concept has been reinvented by many fashion online blogs. The capsule wardrobe must be functional and practical, easing the process of selecting and matching clothing items, but still being able to accommodate all the different types of events during that three months period. That we have knowledge of, there is no application or service capable of simultaneously allow users to create capsules wardrobe and obtain appropriated combinations of clothes based on them.

Our main goal was to develop an intuitive, easy to learn and use mobile application that supports the creation and management of those three month capsules wardrobe. This application also helps users obtain suggestions of complete outfits that are appropriated to a given occasion. Regarding outfit recommendations, our solution accepts textual descriptions in natural language as a possible method to describe desired outfits. Besides capsules wardrobe and outfit suggestions, our solution allows users to store their clothes in virtual closet and simulate the acquisition of new clothing items, showing how many outfits are possible to be created with the new items.

We created an Android application, called Ultimate Capsule, using an iterative and incremental approach. This application was implemented in five modules, contemplating logic for our functionalities, Profile, Virtual Closet, Capsule Wardrobe, Outfit Recommendation and Purchase Simulation, and four algorithms, namely for coordinate clothes, create capsules, interpret users textual descriptions about outfits, calculate most appropriate outfit and simulate the acquisition of a new clothing item.

In the following two sections we will present some background on how to combine clothes in terms of colours and patterns, previous works similar to our solution. Additionally we will present our solution functionalities, development approach and evaluation results. Lastly, on final section we will make conclusions about our work, enumerating our contributions and some examples of possible future work.

BACKGROUND
Before presenting studies related to our work, we found important to understand how clothing items can be put together to achieve good outfits in terms of colours and patterns.

To coordinate outfits is necessary to have in consideration if clothes are in harmony with each or not, in other words, if, according to [3], clothes coordinate well with each other, exists a balance between them. Several aspects can be considered to verify if two clothes are in harmony, namely colour and pattern.

1http://www.wardrobe.co.uk/bio.html (last access on 26/05/2017)

2https://ultimatecapsule.wordpress.com (last access on 11/05/2018)
“Colour is a visual effect resulting from the eyes ability to distinguish the different wavelengths or frequencies of light” 3. Normally, to define colours that result in a harmonious set a colour wheel with twelve different colours 4 is used. Based on colours positioning in colour wheel we can define several colour schemes, in other words, colours sets that create colour harmony. Possible colour schemes are analogous, complementary/opposite, complementary split and triadic scheme. Analogous scheme, sometimes named similar or adjacent, according to [3], means two or three colours that are immediately neighbours of each other. A complementary/opposite scheme, according to [3], corresponds to two colours in opposite sides. In terms of complementary split scheme this is composed by two adjacent colours and one opposite 5. A triadic scheme, according to 3 6, means three colours equally distant in the colour wheel. Additionally, there is a mono tone scheme, called monochrome chromatic scheme, which is constituted by only one colour 5.

Besides colours in colour wheel, there are also colours that are defined as neutrals 5, namely black, grey, white, brown, beige, khaki and navy. These colours are denominated neutrals because they can combine with any colour from the colour wheel. We can define neutral colour scheme, where only neutrals are combined. Similarly to colours in colour wheel, there is also a mono tone scheme with neutrals colours, denominated monochrome tone scheme, composed by only one neutral colour 5.

Furthermore, usually, while combining clothes into outfits, we should not have more than three different colours 7. Sometimes is also defined an accent colour, when one colour from colour wheel is chosen and combined with only neutral colours 7.

In terms of patterns combination, according to [3], patterns should usually be coordinated with solid colours. However, this does not mean that two pattern can not be used together. It is possible to combine two patterns if their colours create harmony and if they have different scales, in other words, if one has small pattern and the other has big pattern 6.

RELATED WORK
In order to understand what has been done and what could be improved, we searched for previous related works. Having in account our main goals, we specifically searched for previous works related to fashion recommendation systems, finding clothes through textual descriptions in natural language and similar applications or services existent that have in account the capsule wardrobe concept, are capable of outfits recommendation and/or have a virtual closet.

Fashion Recommendation Systems
Fashion recommendation systems have the ability of suggesting outfits or clothes pairing which may be interesting to the user. This has been an area more and more explored through the years, arising different systems in terms of approach or conditions considered in the recommendation process. Generally, these systems can suggest outfits appropriate to clothes attributes and occasion, among others.

[9, 13, 19] propose fashion recommendation systems that only consider clothing attributes in the recommendation process and use reference images of complete outfits to suggest their own outfits. In [9] the images are used to suggest outfits similar to them, while in [13] and [19] the images are used to train their own systems. Despite of only consider clothing attributes and require images to train or coordinate clothes, these three systems are similar to our proposal, since all the three systems receive as input a query clothing item and in our proposal we will allow users to obtain outfits recommendations with a clothing item chosen by the user or simulate the purchase of a new clothing item. From these three, the most similar are the systems proposed by [13] and [19] since they are capable of suggest complete outfits, unlike the system in [9], that only suggests top and bottom pairs. The main difference between [13] and [19] relies on the approach followed. Furthermore, all these three systems are identical to our application since they all receive as input an initial clothing item and in our proposal we will allow users to obtain outfits recommendations with a clothing item chosen by the user or simulate the purchase of a new clothing item.

Besides these, exists other fashion recommendation systems that are similar to our solution because, additionally to clothes attributes, they also consider occasion. Some examples of these systems are [2, 5, 17, 11, 12, 15] and [20]. These are more similar to our solution, comparing to [9, 13, 19], because in our solution we have in consideration clothes attributes and occasion in our recommendation process. [2, 5, 12] and [20] are very identical to our solution because they also have a virtual closet. Furthermore, recommendations from systems proposed by [5] and [15] allow to give an initial clothing item that outfit must have, similar to our solution. Additionally, the system proposed by [17] is also identical to our solution in terms of allowing users to textually describe occasion.

Finding Clothes and Outfits Through Natural Language
One of our main goals is to allow users to textually describe, in natural language, a clothing piece that they want to find in their virtual closet and the desired characteristics in the outfit suggested by the system.

As mentioned before, fashion recommendation system proposed by [17], similar to our solution, allows users to textually describe scenario and/or the emotional state that the user wishes to feel with the recommended outfit. Through Open Mind Common Sense [18], a collection of English sentences
about common sense, ConceptNet [14], a tool that relates different concepts with each other, and a handcrafted file with all relations between brands, types of clothing items, materials, some occasions and styles, this system is capable to infer the appropriate style that the suggested outfit must have.

Besides this system, it was also developed systems [21] and [16]. [21] propose a textual and visual clothing search system, that is, a system with two main tasks: (i) given an image of a garment, obtains words or sets of words that describe it; (ii) given a set of words as a description of an outfit, finds an image of a outfit that satisfies the description. The second task of this system is very similar to our main goal, however the several approaches to represent outfits’ images, possible vocabulary and relations between images and vocabulary implemented and compared, besides needing to be trained with thousands of images and textual annotations, are much more complex than our proposal. [16] propose a system that, given a natural language query in English can convert it to SQL and obtain the result of applying it to existing and previously known tables.

From these three systems, the system described in [16] is the most similar to what we accomplished. Similarly to our solution they developed a semantic grammar to interpret textual descriptions given by user to describe a desired outfit. From this interpretation, system [16] is capable of retrieving asked information from the database.

Similar Applications

Services or applications similar to our solution proposal, in other words, with the ability to support the creation and management of capsules wardrobe or recommend outfits, have been created and developed in the past years. That we have knowledge of, the only online service or mobile application been created and developed in the past years. That we have been created and developed in the past years. However, neither of them simultaneously accomplish all our goals. In other words, a mobile application that allows to create and manage capsules wardrobes, suggest complete outfits appropriated to a given occasion, temperature, weather conditions and users’ preferences and past interactions, visualize all the outfits worn or planned to be worn and simulate acquisition of new clothing items. This application will help optimize the process of combining clothes into appropriated and harmonious outfits, complying occasion and users specifications, making this task more simple and satisfying.

SOLUTION

As a final result of this work, our goal was a mobile application, Ultimate Capsule, capable of supporting the creation and management of three month capsules wardrobe and suggestions of complete outfits appropriated to a given occasion and/or other restrictions defined by users. Although we are limiting to a single platform, and there are some disadvantages related to native applications, we implemented this application as an Android native application, since, according to previous works, [4] and [10], a web app would require a constant internet connection (so would require more data consumption, having a negative impact on battery usage), have limited performance and access to device features, and existing platforms for creating a hybrid application still have limitations.

Our application supports both male and female users and they can:

- Upload and characterize all current clothing items into the virtual closet;
- Visualize, edit and remove any clothing item in the virtual closet;
- Create a new minimalist capsule wardrobe for a specific season;
- Visualize all clothing items in some capsule wardrobe;
- Obtain personalized outfits recommendations of bottom and top pieces, or full-body, shoes and optionally accessories, according to colour and pattern matching, occasion and optionally according to a given clothing item or predominant colour;
- Early planning of outfits;
- Save and visualize outfits worn or to be wore in a specific date;
- Simulate purchase of new clothing items.

With the intention to develop a solution that supports these tasks, we created and developed five modules and five algorithms. Four of our five modules store the corresponding logic for each main functionality, namely
Virtual Closet, Minimalist Wardrobe Capsule, Outfit Recommendation and Purchase Simulation. The fifth module, Profile, correspond to profile functionality logic, created with the goal to obtain users information and provide a more personal experience. Concerning algorithms, we created ClothesCoordination, CapsuleCreation, FashionNaturalLanguage, OutfitRecommendation, and SimulatePurchase. Additionally, we have created a SQLite database where we store all users’ clothes, capsules wardrobe created and outfits worn and to be wore. This database is locally stored in users devices.

Profile Functionality
In our solution we depended on some users information, more precisely gender and location for, for example, determine clothes categories and seasons start and end dates. Having this objective in mind and the purpose to support a more personal experience, we added a profile functionality. In this context, we allow users to create, see, edit and delete their profiles.

In our application, users are characterized by name, date of birth, gender and location. The first two, name and date of birth, are not entirely needed in our application, because we do not influence our solution functioning by them, but we decided to have them since it could be helpful in the future, for example in our recommendation or simulation algorithms. At this moment, the only informations that can influence our application are gender and location. The former, gender, has impact on clothes categories considered by our application, while the latter, location, determine each season start and end dates. Regarding location, we decided to ask users in which hemisphere they are, instead of country. We opted by this, since we only really needed to know in which hemisphere users are and presenting the entire list of countries, even if they were alphabetically ordered, would lead to a very exhaustive list of possible countries, being very inefficient, time wasting and frustrating to users, and also would imply the implementation of a function that matched selected country with correspondent hemisphere.

Virtual Closet Functionality
In order to create minimalist capsules wardrobe and give outfits recommendation with users’ clothes, we had to acquire knowledge about them, meaning we needed to save each clothing item information. In other words, we needed to support the creation and management of a virtual closet, so we added the Virtual Closet functionality to our solution. In this, users are able to see all their clothes and add new, modify or remove existing ones.

The users’ clothing items in the virtual closet were defined by their own image, type (and category if applicable), colour, type of pattern (and pattern scale if applicable), season and occasion. In terms of clothes images we offer two options to users: the possibility to take a photography of the clothing item or select a image from device’s gallery. In terms of clothes types, either it is male or female user, we considered top, bottom, full-body, shoes and accessories. We did not consider workout a clothing type since there was no consensus between previous works and applications analysed. In terms of clothes categories, we have in account if it is a male or female user, having different possible categories for each one. Both clothes types and categories choices were based on previous works and similar applications and also some research on online shopping stores. We do not allow clothes types and categories customization, like some similar applications, because these are used in our recommendation algorithm while calculate appropriate outfits. Regarding colours choices, we had in account the twelve colour wheel that is normally used for combine clothes, as well as neutrals colours, mentioned in Background section. In terms of patterns, we considered the ones more frequent in the links mentioned in Background section. We decided to additionally have a pattern option named other, with the intention to support all patterns that may exist but were not considered as more commons. Concerning patterns, we additionally defined the attribute pattern scale, as small or big, to all patterns. Regarding clothes occasions, similarly to clothes types and categories, we base ourselves on previous works and similar applications and also some research on online shopping stores. In our solution, each clothing item can have more than one occasion associated. In terms of seasons, since the capsules wardrobe created by users are valid for three months, we considered spring, summer, autumn and winter. Just like occasions, users will have the option to choose several seasons for each clothing item.

In figure 1, we can observe our virtual closet design while visualizing “Shirts and Blouses” already inserted.

![Figure 1. Virtual closet overview when visualizing “Shirts and Blouses” already inserted.](image)

Capsule Wardrobe Functionality
As mentioned before one of our main goals was to support the creation and management of three month capsules wardrobe, so we created the Capsule Wardrobe functionality. In this, users are able to see all their capsules and also create new, edit and delete existing ones. Besides, since we allow to have multiple capsules for the same season, users can select which capsule should be considered as current for the present season.
The capsules in our solution are characterized by name, season, start and end dates and number of clothing items by type. Since in our solution we allow capsules with the same season and number of clothing items, we needed an unique attribute. With the intention to generate capsule between capsule with same season and number of season, we added the name attribute to capsules. Concerning season, this was defined because our main goal from start was to allow create capsules dedicated to each dates of the seasons. We chose the last option, because our main goal from start was to allow create capsules dedicated to each dates in other words, with no more and no less than three months duration. With a little research, we come to realised that, from all possible ways to define each season start and end dates, there were two that are more commonly used in real world. These two are: (i) meteorology method, where seasons start and end days always correspond, respectively, to the first day and end day of specific and clearly defined months; (ii) astronomic method, where seasons are determined by equinoxes and solstices days. We decided to chose the meteorology method, over astronomic, since the former only required to know if users live in north or south hemisphere, while the latter would required to know both the country where user live and equinoxes and solstices dates for current year and for selected country, since they vary from year to year, depending on location. In respect to capsule clothes, we also had two possible scenarios, ask users the total number of clothes and we define how we divide this total into the several types, in other words, which are tops, bottoms, full-body pieces, shoes and accessories, or ask users how many clothing pieces they want of each type. We opted for the last scenario, since from the research made we observed there are no consensus about which clothes actually count to total number of clothes in capsules neither about how to divide this total for the several clothes types. So, we allow users to customize the number of capsule clothes in terms of tops, bottoms, full-body pieces, shoes and accessories, ranging from zero to the number of existing clothes for each type.

From information specified by our users about new capsules, more specifically season and number of capsule clothes, our solution is capable to create capsules with clothing pieces that maximize the number of possible outfits, complying with number of pieces asked by users. When season or number of pieces of a existent capsule changes, our solution recalculates capsule’s clothing items.

**Outfit Recommendation Functionality**

Similarly to the module Capsule Wardrobe, in order to accomplish the goal of give outfits suggestions appropriate to occasion and users’ restrictions, we added the Outfit Recommendation functionality. In this, users can obtain an outfit suggestion for a specific day, through textual description or attributes selection. Regarding attributes selection, we allow users to specify occasion, clothes types of the outfit, one specific clothing piece that the outfit must have, dominant colour and if it should have accessories or not. With information specified by our users, through textual description or attributes selection, and clothes from current capsule wardrobe, or, if there is none defined, from virtual closet, we obtain a list of appropriated outfits, from which we choose one. From the suggested outfit, users can accepted and save it to a database or ask for another suggestion with same characteristics. The suggested outfit saved into database for current day or selected day can be seen in the initial screen.

In figure 2, we can observe one example of outfit recommendation result when specified white blouse as specific piece selected.

**Simulate Purchase Functionality**

Identically to Capsule Wardrobe and Outfit Recommendation, to achieve our goal of helping users to buy only clothing items that combine with maximum number of clothes possible, our solution has a functionality where users can verify how a new piece, that they are considering to buy, combines with their virtual closet clothes. In other words, we created the Simulate Purchase functionality. In this, through a process similar to adding a new clothing piece into virtual closet, users can observe the number of outfits possible with this new piece and a small sample of those outfits. Considering these informations, users are more capable of making a conscious decision about buying (or not) a certain clothing item. With the intention to avoid duplicate and redundant actions, at the end of this process, we allow users to add the simulated piece into the virtual closet.

In figure 3, we can observe possible outfits with new denim jeans, after simulating its purchase.

**Clothes Coordination Algorithm**

In the CapsuleCreation, OutfitRecommendation and SimulatePurchase algorithms, we needed to know...
if two clothing items combine or not, considering their colours and patterns types and scales. With the intention to avoid duplicate code through these three algorithms, we abstracted this task into one algorithm called ClothesCoordination. In this, we implemented the rules mentioned in Background section used normally to combine clothes in terms of colours and patterns. To abstract these rules and allow to be used by other modules, we implemented two functions that tell us if two clothing pieces combine in terms of colours and patterns, respectively functions clothesPatternCoordination and clothesColoursCoordination.

Capsule Creation Algorithm
In our Capsule Wardrobe functionality, to abstract our capsule creation methodology, we implemented an algorithm, called CapsuleCreation, which is capable of choosing the clothing pieces that maximize the number of possible outfits, complying season and number of clothes by type. The result of this is a list of clothing items that constitute a capsule.

Our first step in our capsule creation algorithm is to obtain all clothes, by type, for a specified season. The second step corresponds to match clothes and chose the ones which maximizes the number of possible outfits. This is accomplish by iterating through two lists of clothing items and registering the number of possible outfits with each clothing piece from the first list, using the algorithm ClothesCoordination to see if two pieces combine or not. This information is stored in a map of integers keys, corresponding to clothing items ids from first list, and integers values, corresponding to number of possible combinations for clothing item with id saved in key. At the end, to decide which clothes coordinate best, we order the map by value, by reverse order, and chose the first \( n \) clothes, being \( n \) the selected number of clothes for that type by user. We defined that for coordinating clothes we would first calculate tops and bottoms, then shoes, full body pieces and finally accessories. We chose to start from tops and bottoms and then shoes, instead of full-body pieces first, since this way we could know which shoes go best with tops and bottoms. After selecting tops and bottoms, we repeat this process to remaining types, considering now the clothes stored already as capsule clothes, instead of clothes retrieved from database to specified season. This algorithm stops when all capsule clothes types have been calculated, returning a list of clothing items.

Fashion Natural Language Algorithm
One of our goals as functionalities in our application was to be able to process textual descriptions in natural language regarding outfit suggestions. In other words, our goal was to be able to, given an outfit description in natural language, transformed it into a useful structure that we can used to calculate the most appropriated outfit. In order to accomplish this we implemented an algorithm called FashionNaturalLanguage, which corresponds to a natural language parser related to fashion, which input is a natural language description of an outfit and the result corresponds to pairs of attribute values predefined by us. In these, we store outfit general attributes, namely dominant colours, patterns and/or appropriated occasions, as well as outfit specific pieces descriptions.

To implement this algorithm, we needed to have a development corpus, in other words, a set of phrases that exemplified users descriptions of outfits. Our development corpus was created by asking to various users to pretend to have different occasions or at morning, choosing their outfits, and write phrases that would describe the outfits they would imagine or like to receive, to these imaginary scenarios. Since we did not intended to have synonyms or functions to relate unknown words in our context with words in our context, we asked users to use the clothes types and categories, colours, patterns and occasions defined by us in our application. At total, we collected 135 phrases to our development corpus. Some example of phrases from our corpus are: “formal outfit”, “jeans with t-shirt” and “casual outfit mainly with white shirt”.

Our algorithm, FashionNaturalLanguage, is constituted by four stages: pre-processing phase, mentioned entities recognition, syntactic and semantic analysis.

In the first stage, pre-processing phase, we receive the textual description about an outfit and we lower case all words, substitute ; with line break, substitute symbols & and / with, respectively, “and”, “or”, remove punctuation as well as stop words. In terms of stop words, we used the ones defined in \(^\text{19}\), except “and”, “or” and “with”. We decided to not remove this stop words from input phrase because they were relevant to us while parsing phrases, according to grammar created by us. Additionally to these commonly used stop words, we add our own stop words based on the sentences collected. More specifically we defined as stop words: “night”, “day”, “small”, “matching”, “big”, “colourful”, “dinner”, “piece”, “pieces”, “outfit”, “outfits”.

In the second step, mentioned entities recognition, given our textual description without punctuation or stop words, we
used an implementation of Aho-Corasick \(^{20}\) to find words related to clothes types and categories, colours, patterns and occasions and replace them by corresponding tags. We used Aho-Corasick because, citing [1], this is a “simple, efficient algorithm to locate all occurrences of any of a finite number of keywords in a string of text”. While recognizing mentioned entities from one input sentence, we returned two separate phrases, one phrase with keywords replaced by correspondent tags and a second phrase with each tag value surrounded by square brackets.

After having our keywords identified, to interpreter textual description of an outfit given by users, we needed to analyse it syntactically. To do this, we developed our own syntactic parser. We opted to develop a semantic grammar, over more recent technologies like neural networks, because our corpus was very small. According to [7], a semantic grammar corresponds to a grammar specifically develop to a certain defined domain, so, in this case, a grammar specific to describing outfits. We had several iterations though our corpus phrases to created our grammar. To implement our syntactic parser, based on our semantic grammar, we opted by a bottom-up parser, because we had a very small corpus and our grammar was very simple. We were able to implement all our grammar rules with the exception of defining an outfit as a set of clothing pieces followed by outfit general attributes. To store information read with syntactic parser and know when to stop we used a stack.

As a final step, to comprehend the semantics behind the textual description given by users, we needed to add some semantic to encountered tags and syntactic structures. As a simplification we opted by joining semantic analysis to the syntactic parser. To perform this we added a second stack to our parser, where the correspondent values read are stored as well as objects that correspond to non terminal symbols in our grammar.

Through these fours steps, our algorithm is capable of understanding for example “White date outfit with Dress”, a phrase from our development corpus, and return a set of pairs attribute value defined by us. In this case, we end up with white stored as outfit dominant colour, date as outfit occasion and dress as a clothing item that must constitute our recommended outfit. This information is then used by OutfitRecommendation algorithm to calculate the most appropriated outfit.

**Outfit Recommendation Algorithm**

Another of our main goals was to calculate the most appropriated outfit for specified requirements by users, so we implemented an algorithm called OutfitRecommendation. This is subdivided into two, depending if recommendation is through natural language textual description or attribute selection. In the former scenario, natural language textual description, the first step correspond to call FashionNaturalLanguage algorithm to interpret textual description, and then, based on result calculate the most appropriated outfit. In the latter scenario, attributes selection, the algorithm based on required attribute occasion and optional attributes outfit pieces types, specific piece, dominant colour and if it should have accessories or not, calculate the most appropriated outfit.

Through attribute selection, users must select occasion and can, if they want, personalize outfit regarding pieces type, specific clothing piece, dominant colour and if outfit should have accessories or not. In this case, in our algorithm, we verify these attributes and calculate outfits based on them. In terms of occasion, since this is required we always know to which occasion the user wants to wear the outfit to. So, we are able to only consider clothes that have the chosen occasion as one or only occasion possible to wear them. Regarding optional arguments we needed to take into account scenarios when they were defined as well as undefined. Concerning outfit pieces type optional argument, it allow users to specify if they desire an outfit with top and bottom pieces or an outfit with a full-body piece. So, this restricts which clothes types we try to combine. Regarding specific clothing piece, this attribute allows users to select a specific clothing item. So, if defined we try to complete outfit with missing clothes, otherwise we start to combine outfits without any specific clothing item from start. In terms of dominant colour, we allow users to select one colour to be present in our outfit, in other words, if this attribute is defined, we try to have at least one piece with the chosen colour. Since we defined this attribute as dominant, we considered that this means the top, bottom or full-body piece in our recommended outfit should be this colour, or if piece has multiple colour, this should be one of them. If this argument is undefined we disregard colour as a requirement to create outfits. Respecting to have accessories or not, this determines if we should try to outfits with or without accessories. From these optional attributes, users can customize all, some or none of these, with the exception of having accessories, that if users do not customize its default value is “No”. If none of these optional attributes are customized, our outfit recommendation algorithm try to calculate outfits only considering occasion as requirement. After calculating possible outfits, complying occasion and optional attributes, outfit pieces type, specific clothing piece, dominant colour and if outfit should have accessories or not, we obtain an list of outfits. If this list is not empty and there is more than one possible outfit, we randomly select one. If there is only one, that is the outfit returned. If the list is empty, our algorithm returns an empty list, which is then interpreted, by our OutfitRecommendation functionality, as not existing an appropriated outfit.

Concerning finding an outfit described through natural language, our first step corresponds to call our natural language algorithm, FashionNaturalLanguage, to interpret textual description. If our natural language algorithm is not capable of interpret the description given by users, it returns null and OutfitRecommendation algorithm immediately stops. Otherwise, if it is able to understand description given by user, it returns a set of pairs attribute value with all characteristics related to outfit in general, as well as outfit pieces descriptions. Regarding outfit pieces, if there
is no information about what pieces should constitute outfit, we try to match all clothes. Otherwise, we try to complete the outfit, similarly to outfit recommendation through attribute selection when there is a specific clothing item selected. In this case, since we can have more than one piece specified, we first obtain from database all described pieces and only them, considering the missing ones, complete the outfit. After having calculated possible outfits with pieces described or, if there are no pieces descriptions, after matching all clothes retrieved from database, we check which ones comply with outfit general attributes mentioned, if they exist. Concerning outfit general attributes, having in account the grammar defined by us, users can customize outfit occasion, colour and/or pattern. Identically to attribute selection recommendation method, after calculating possible outfits regarding pieces described and outfit general attributes mentioned, we obtain an list of outfits. If this list is not empty and there is more than one possible outfit, we randomly select one. If there is only one, that the outfit returned. If the list is empty, our algorithm returns an empty list, which is then interpreted, by our Outfit Recommendation functionality, as not existing an appropriated outfit.

**Simulate Purchase Algorithm**
Additionally to capsules creation and outfit recommendation, our goal was to help users decide if they should buy a clothing piece or not based on possible outfits with it. To calculate possible outfits with specified clothing item, we created an algorithm called SimulatePurchase. This, with knowledge of new clothing piece type, category, colours, pattern type (and scale if applicable) and appropriated seasons and occasions, determines possible outfits with specified item. This algorithm works similarly to OutfitRecommendation algorithm when user selects a specific clothing item, with the exceptions that SimulatePurchase returns all possible outfits and, contrarily to OutfitRecommendation that allows users to determine if it should have accessories or not, this returns always outfits with accessories.

**DEVELOPMENT**
Our intent was to build an application with a graphical user interface intuitive, easy to use and learn. In order to explore and compare different design alternatives, in a short period of time with a low modification cost, we followed an iterative and incremental design approach. With this we started by creating some storyboards of the main tasks, in order to define the different states of our application, and then, incrementally, we added more complexity, going from a low-fidelity and non-functional prototypes until a final product. More specifically we have:
1. Defined main tasks;
2. Created storyboards for them;
3. Created low-fidelity non-functional prototypes (paper prototypes);
4. Evaluated low-fidelity non-functional prototypes;
5. Created functional prototypes;
6. Performed a mid evaluation on our functional prototypes;
7. Improved our functional prototypes considering feedback from mid evaluation;

The first step in our solution development process consisted of defining main tasks and creating storyboards to them. We started with storyboards, since, according to [6], they can show the sequence of steps required to support some tasks, allowing to see and understand how our application will flow and react to users’ interaction. We consider that the most important tasks that our solution would support were: upload a new clothing piece, create a new capsule wardrobe, obtain an outfit recommendation and simulate purchase of a new clothing item. We consider these because are the most critical: either they are more complex or more commonly performed by the user.

As a second step in our solution development process we created low-fidelity and non-functional prototypes. Since our main goal was an android application to be used in daily life of our users, we choose to use a smart phone device frame. We started by designing our side menu and home screen low-fidelity and non-functional prototypes. After those, having in account our main tasks defined while designing storyboards, we have chosen to implement four of our five main functionalities, namely Virtual Closet, Capsule Wardrobe, Outfit Recommendation and Purchase Simulation. More specifically, we decided to implement one example of each main task associated to these functionalities. In this stage, having in account that we wanted our application to be available to everyone, independently from where users are, we decided to design our low-fidelity and non-functional prototypes in English.

After having created and evaluated the low-fidelity non-functional prototypes, we started to develop our functional ones. Considering that our goal was to create an intuitive, easy to learn and use application, we chose to start implementing interface related to our solutions functionalities iteratively, and only after having part of it implemented, started to create our solution algorithms.

As a final step of our development process we evaluated our solution in terms of usability and utility as well as our fashion natural language algorithm in terms of accuracy.

**EVALUATION**
Our solution evaluation was consisted by two different kind of evaluations. One for evaluating our solution interface in terms of usability and utility and other for assess our developed natural language algorithm, FashionNaturalLanguage.

**Solution Interface Evaluation**
Concerning usability and utility, with the intention to obtain early feedback from the users about how to build our application in an intuitive way, easy to use and learn, and reduce the number of modifications in our final application, we had three phases of evaluation. Two were during development, evaluation of our low-fidelity non-functional prototypes and formative evaluation of our functional prototypes, and the third one
was after finishing our solution, to evaluate sumatively it in terms of usability and utility.

During both low-fidelity non-functional prototypes and formative evaluation of our functional prototypes, we asked users to perform some tasks, while thinking aloud, and classified these tasks in terms of easiness. Additionally, in the end of the evaluation we asked users general opinions about our interface. Based on users’ thoughts and opinions, as well as tasks classification, we were able to see which aspects of are more difficult or less intuitive to understand in terms of users’ perspective, in other words, we were able to find possible improvements to be made in our application interface. Besides evaluating our solution interface, in formative evaluation of our functional prototypes, with the intention to build an evaluation corpus to our natural language functionality, we additionally asked users to give us ten sentences describing outfits, giving them the same rules that we had given to users while creating development corpus.

In our final solution regarding usability and utility we conducted two different tests, usability tests and cases studies. We choose both because we know, from [6], that a good solution must have high usability and high utility and these these two tests, complement each other, since it allow us to assess both usability and utility. In this final evaluation stage, since we wanted to clearly evaluate our solution efficiency, effectiveness and user satisfaction, we chose to collect data from each test performed and treat them statistically, using direct observation. We chose to collect time taken to complete each task, number of mistakes performed and quantity of tasks completed successfully, and treat them statistically, since, according to [6], these are the most commonly used. In order to correctly collect these data, we had to clearly each task and its correct end. Besides these measurements, that allows us to assess mostly efficiency and effectiveness, we also asked ask users to answer a usability quiz, with the intention to evaluate their satisfaction with our solution. To make sure our tests were all performed under the same conditions, we had previously defined a usability test script. From collected data, more precisely number of tasks successfully finished, time spent performing each task, errors made, tasks classifications and users satisfaction, we can conclude that there are room for some improvements, since some mistakes were made by users and some tasks were not classified as easy or very easy by all users. However all tasks were successfully finished by all users, users were able to recover from errors and overall users satisfaction was relatively high, about 83.54. So, in our opinion, we consider that we were able to create an intuitive, easy to use and learn solution that also allow users to recover from errors made. Besides usability tests, we conducted three cases studies in order to assess our solution utility. In these tests, we asked users to perform the same tasks of usability test, while thinking aloud. Our goal with these tests was to acknowledge our application utility, in other words, real users opinions about our application while interacting with it. With this goal in mind, we chose users that corresponded to our users target group, users interested in fashion related application and/or with necessity of an tool to help them organize their closet into capsule wardrobes and/or reduce the task of finding an appropriated outfit for everyday. In case studies, users expressed that although they might not use the application on a day-to-day basis, it would be “useful specially to create new and unpredictable combinations”, also to “plan outfits for travels” or “when is necessary a more formal or specific kind of outfit”. They also said that allowing to chose a certain piece while recommending outfits were helpful since “sometimes the major difficulty is to find outfits with a specific piece that we want wear”. Additionally, users though that the possibility of see how a certain new piece can coordinate with users’ clothes is “very useful because helps buy only clothes that match with a high variety of clothes” that they have.

**Fashion Natural Language Algorithm Evaluation**

Through the phrases collected while performing formative usability tests we were able to create an evaluation corpus to evaluate our fashion natural language algorithm with 50 phrases. To perform this evaluation, without having to interact with our application interface we build an external project with natural language code from our solution. By doing this, we could create a text file with all sentences that constitute our evaluation corpus, as well as functions that read this file content and outputs how many phrases were parsed and how many were not, showing the ones that were not able to interpret.

From this evaluation, we could observe that our natural language algorithm has 58% accuracy rate, in other words, were capable of correctly interpret 29 of 50 phrases. Some examples of correctly interpreted evaluation sentences were “Casual floral”, “Party outfit with plain sleeveless shirt, red shorts and beige shoes.” and “Casual outfit plain black white”. From the failed sentences, we concluded that about this rate could be improved if we were able to implement our grammar rule where we define an outfit as a set of pieces followed by outfit general attributes, if we added more our own stop words and if our grammar was capable of recognizing scenarios where some attributes regarding clothing items are to right of type/category.

**CONCLUSIONS**

Motivated by the capsule wardrobe concept and the weaknesses of previous works, we created an Android application, called Ultimate Capsule, that allows users to store their clothes, create three month capsules, obtain outfits recommendations according to several conditions, view the outfits worn in the past days and simulate the purchase of new clothing items. Besides that, we also support colour and pattern combinations rules and textual descriptions in natural language to find outfits. This application was created by following an iterative and incremental approach, with a SQLite Database, five modules and five algorithms. Each module corresponded to one functionality logic and as algorithms we implemented ClothesCoordination, CapsuleCreation, FashionNaturalLanguage, OutfitRecommendation, and SimulatePurchase.

To create our application through an iterative and incremental approach we started by defining the most critical tasks
and sketching storyboards for them. After this, we created low-fidelity non-functional prototypes and evaluated them. Through users feedback, we were able to improve less intuitive aspects of our application and implemented a first version of our functional prototypes. With these, we then performed a formative evaluation, where we were able to check problems in our application, which we improved and corrected. Having our functional prototypes fully implemented, we evaluated it regarding usability and utility, through usability tests and case studies. From usability tests, more precisely collected data and statistical study made on them, we observed that there is room for improvement, however all users successfully finished all tasks, were able to recover from errors and overall users satisfaction was relatively high, about 83.54. Similarly, from case studies, we concluded our solution is useful for our users, since it helps them to plan unusual combinations, for specific occasion or with a certain piece, plan ahead and also see if how a new clothing piece combines with remaining clothes. Additionally, we evaluated our natural language algorithm in terms of accuracy with our evaluation corpus. In other words, we calculated the percentage of how many phrases were correctly processed by it from a set of evaluation phrases. We could conclude that our natural language algorithm has a 58% accuracy rate, however, by observing failed phrases we realised that can also be improved.

With this development approach and evaluation phases, we were able to create an intuitive application, easy to use and learn, that not only allows users to create a virtual closet and organize it into capsules wardrobe, but also eases the process of learning, that not only allows users to create a virtual closet and organize it into capsules wardrobe, but also eases the process of learning, that not only allows users to create a virtual closet and organize it into capsules wardrobe, but also eases the process of learning, that not only allows users to create a virtual closet and organize it into capsules wardrobe, but also eases the process of learning.

Contributions

Our main contributions with this work were:

- An algorithm, ClothesCoordination, that determines if two clothing pieces combine in terms of colour and pattern;
- An algorithm, CapsuleCreation, capable of automatically create a three month capsule wardrobe;
- An algorithm, FashionNaturalLanguage, capable of process and interpret an natural language textual description about an outfit;
- An algorithm, OutfitRecommendation, with the ability of automatically recommend an appropriated outfit, from the users' clothes in current capsule or virtual closet, according to given requirements by user, like occasion, outfit pieces and/or dominant colour;
- An algorithm, SimulatePurchase, that allows to simulate the acquisition of a new clothing item, showing possible outfits with that piece;
- The creation of an intuitive, easy to learn and use Android mobile application, called Ultimate Capsule, that merges outfit recommendation system and natural language.

Future Work

We were able to accomplish our goals, however there are some possible improvements that can be done in our work. More specifically, some examples are:

- Automatically characterize clothing items through their images, obtaining more information about clothes, for example shaping and texture/type of fabric;
- Improve ClothesCoordination to have in consideration clothes texture/type of fabrics and shaping;
- Improve CapsuleCreation algorithm to allow users to personalize clothes in terms of overall capsule dominant colour or colour pallate;
- Extend FashionNaturalLanguage to Portuguese language;
- Improve FashionNaturalLanguage by implementing it through more recent technologies, like neural networks;
- Add more information to OutfitRecommendation algorithm, for example temperature and weather conditions, with the intention to provide the most appropriated outfit possible.

REFERENCES


