Framework for Recommending Clothes Based On Convolutional Network Derived Stylespaces

Stefan Kanan
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius
Skopje, Macedonia
kanan.stefan@students.finki.ukim.mk

Ilinka Ivanoska
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius
Skopje, Macedonia
ilinka.ivanoska@finki.ukim.mk

***Abstract*—Over the years the internet has taken a very central role in our society. While it might have been unthinkable in the past, many people today elect to read their news, do their shopping or subscribe to various services online. This has made it increasingly more important for sellers to offer the right ad, relevant to our interests. In this paper we investigate a framework for matching clothing products based on their stylistic compatibility. The algorithm works by using Convolutional neural networks to find an appropriate embedding, that is close to other stylistically compatible products. We create a framework in order to test the efficacy of this algorithm. Our results show that the framework is capable of recommending matching clothes better than those chosen at random. We also visualize our items using tSNE and compare our plot with that of the original paper.**

***Index Terms*—Siamese Networks, tSNE, Recommendation algorithm, Neural Networks**

I. Introduction

Today, more than ever advertisements are an unavoidable part of daily life. Whether reading an article, watching a video or simply opening an app on our smartphone, it seems like our gaze simply cannot escape the ubiquitous ads. Given our increasing fatigue, it is ever more important that advertisers wisely choose what to offer lest they lose our fleeting interest. This has brought in the age of tailored advertisement. Whether entertainment, vacation spots, hotels or clothes, large and complex algorithms are hard at work crunching data, to offer us suitable recommendations.

In this paper we implement the recommendation framework described in [6], which uses a Siamese network in order to pair items based on their compatibility. To that end a database is build, in which we store our items and a pipeline for recommending suitable clothes given some prior user specified clothing item is created. We conduct a user study where users have to choose whether the clothes they have been recommended by the algorithm are in fact a suitable match and see whether the algorithm is better than choosing items at random.

The algorithm works by translating the image of the apparel into a new style space with the aid of a neural network. This network is first created by supplying it with both compatible and incompatible pairs of clothes, with the hope that it will learn some notion of style with which it would be able to discern matching and non-matching items. This new style space brings items that go well together close to each other and splays items that do not further apart. Thus, for any given image, the algorithm can recommend an item it sees as the most compatible to the one given by the user.

The neural network assigns an embedding to each item, which describes its location in this new style space. Naturally items that go well together are expected to be close to each other and vice versa. Knowing this it would be beneficial to visualize the style space. However, the high dimensionality of the embeddings makes this unfeasible using traditional techniques. One way in which to express high dimensional data is using he tSNE visualization algorithm, which we will use here, that expresses high dimensional data in terms of the joint probability between nodes.

Our paper is structured as follows. In Section II we describe our dataset and all of the pre-processing steps as well as introduce the concept of the Siamese convolutional networks. In Section III we offer an implementation of [6] which is explained in detail. In section IV the tSNE algorithm is explained and we present the results of this algorithm on our data. In Section V we conduct a user study and discuss the results. Finally, we offer a conclusion.

II. Materials and Methods

A. Dataset

Our data consists of clothing products. Each product a contains: an id and a link to the image of a: I_a, additionaly some items contain a $description$. Part of the recommendation algorithm is matching an item to a target category (wristbands, dresses and etc). Our dataset doesn't contain a dedicated category attribute, yet the description part contains valuable information about the category the item belongs to. In order to extract the descriptive attribute into its corresponding categories, we first clean and tokenize this attribute for each item in our dataset. After doing so we count the appearance of each category in our dataset, then we pick manually the more commonly found categories while excluding artifacts, brands and other non-valid categories. Finally, we end up with a handful of basic clothing categories (such as pants, shirts and jackets). A caveat concerning our descriptors, is that in parsing this attribute rather than having a dedicated $category$ attribute we sometimes end up with the wrong categories for

our items (ex. a belt whose description may say "belt for jeans" but which because of the parsing process may end up in both the belt and the jeans category). In order to avoid this problem we suggest a more intelligent parsing algorithm or manually classifying items into their suitable categories. Another problem we ran into with our dataset, was duplicated data. We removed these duplicates on the basis of having the same image, however in doing so we ended with 7495 items from our original 26375 ones, this is in comparison with the nearly two hundred thousand images in [6].

B. Methodology review

The main idea behind [6] is the use of Convolutional neural networks to learn a style space so that clothes that stylistically go well together are part of the same style space, while clothes that are not compatible, are not.

This is done by utilizing a Siamese neural network, trained on mini-batches of 1 compatible pair of clothes $(a, b)^+$ and 16 negative pairs $(a, b)^-$ [6]. These pairs are heterogeneous dyads, which means that the two items come from different clothing categories. In doing so, they argue, we force the network to learn to match items from different categories, rather than group similar items together.

The Siamese neural networks are a specie of Convolutional neural networks consisting of two identical neural networks sharing weights θ. During training the neural networks are fed a pair of items and the desired output y, denoting whether a pair is compatible or not. The result is then used in a contrastive loss function [1], which calculates the error and is used to train the network:

$$L(\theta) = \sum_{x_q, x_p} L_p(x_q, x_p) + \sum_{x_q, x_n} L_n(x_q, x_n) \tag{1}$$

Where the first term penalizes compatible images that are found to be far apart and the second term gives the penalty for incompatible images that are nearby.

$$L_p(x_q, x_p) = \|x_q - x_p\|_2^2 \tag{2}$$

$$L_n(x_q, x_n) = \max(0, m^2 - \|x_q - x_n\|_2^2) \tag{3}$$

In our case, given two items and their images $a : I_a$ and $b : I_b$ we want to translate these images into their stylespace or rather find their embeddings $F : I_a \to s_a$ and $F : I_b \to s_b$. This comes down to finding suitable descriptors which are either distant if the two items are not compatible or close to each other if they are. [6] uses a GoogleNet network, pre-trained on the ILSVRC image dataset [4]. The original intention of which was to correctly classify images. The Siamese network is then build by removing the last softmax layer, and replacing it with a 256 node fully connected layer. [6] [1]

The network is now fine-tuned using pairs from different categories in order to force the network to learn style compatibility rather than classifying items as near when they are from the same category. These pairs make use of the 'items frequently brought together' section on the Amazon dataset. [2] [3]

Finally, in order to recommend an outfit the algorithm clusters items from the same category. In the paper this is done using the k-means algorithm with 20 clusters per category in order to lessen the calculation when comparing items. Then our query item finds the nearest centroid and is compared with items in that cluster. Finally, we use a knn algorithm with $n = 5$ to find the 5 closest items and from them choose the closest one to show.

III. Implementation

Our implementation uses the pre-trained network[1] in [6]. We also build a database in which to store our items and embeddings. Because each item a can be part of many categories $a \in C_i$ where $i = 1...n$ and many clusters, we deemed that a relational database would prove unsuitable given the structure of the data, and would provide worse performances as the amount of items increased. Therefore we elected to use Neo4j, a graph database that stores data in the form of nodes (which can have one or more roles), edges (representing relationships) and attributes. Neo4j has the added benefit of constant performances regardless of the amount of data. A graph database is uniquely equipped to deal with items that can be part of many categories whilst keeping the model clean resulting in simple queries. A model of our database can be seen in (fig 1). Next we run our Siamese network to extract the embeddings from our images, using each category and gender we then cluster our items. Our implementation tends to keep the recommended 20 clusters per category, except for less numerous categories where we use 10 clusters, and for categories with 50 items or less which are not clustered.

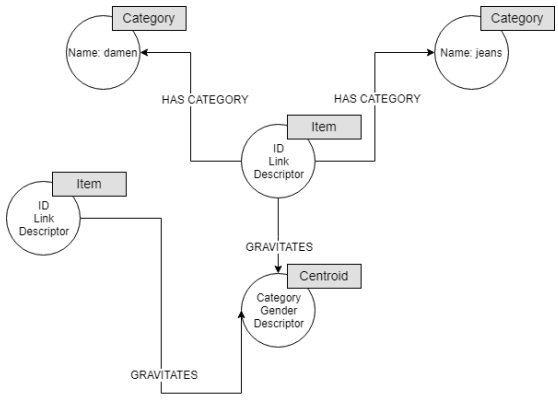

Fig. 1. The Neo4j model

Finally, we build a pipeline where a user can provide either an id or a link and a target category and gender and can be recommended compatible items. We do this by using a knn algorithm which finds the 5 most compatible items. Even

[1]The network, implemented in Caffe, can be found at https://vision.cornell.edu/se3/projects/clothing-style/

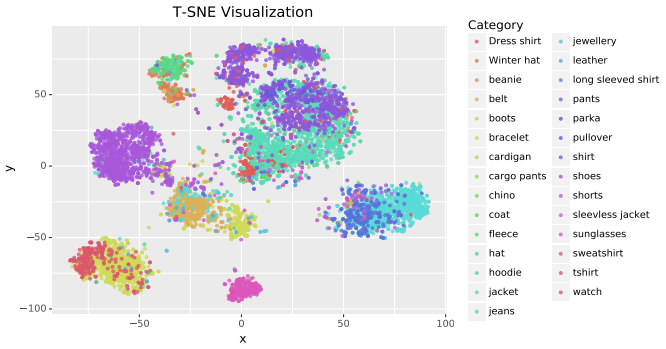

Fig. 2. Visualization of the item stylespace using tSNE, $Perplexity = 50$, 5000 iterations

though [6] recommends clustering the items and finding the knn for our item in part to avoid presenting the user with items from the same category as our query item that have been miscategorized, we found that this was frequently the case in our pipeline. We hypothesize that the network still classifies items from the same category as being similar, perhaps a different training regiment may solve this issue, however more research is needed on this topic.

IV. T-SNE Visualization

As in the original paper, after finding the descriptors for all of our items, we visualize them using the tSNE algorithm. tSNE (Short for 't-Distributed Stochastic Network Embedding') is a technique for visualizing highly dimensional data which works by converting the original data $x_i \in X$ for $i = 1, 2...n$ (each x_i is a row of variables in some dataset X) into two or three dimensional coordinates fit for visualizing, this latter set is labeled as $y_i \in Y$ for $i = 1, 2...n$, called a map. [5] The plain SNE algorithm first converts the euclidean distance between points into conditional probabilities, and then tries to find lower dimensional points whose conditional probability matches the one on the high dimensional data. The conditional probability of x_i is

$$p_{i|j} = \frac{\exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq j} \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}\right)} \tag{4}$$

.

With an unknown variance σ_i^2, the variance is found by using

$$Perp(P_i) = 2^{H(P_i)} \tag{5}$$

Where $Perp$ is the perplexity or the approximate number of neighbours of each node, and is given by the user, and $H(P_i)$ is the conditional entropy

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{6}$$

.

Correspondingly, the conditional probability of the lower dimensional representation of our data, is

$$p_{i|j} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq j} \exp\left(-\|y_i - y_j\|^2\right)} \tag{7}$$

with a fixed variance of $\sigma^2 = \frac{1}{2}$, this scales the map, but in doing so loses some detail of the original data, for there, every point had its own variance. In order to find suitable lower dimensional data points, we minimize the Kullback-Leibler divergence

$$C = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \tag{8}$$

The cost function is then minimized by using the gradient descent method. However this cost function of the plain SNE algorithm can often times end up in a local minima, requiring multiple runs or other optimization. Because of the way the Kullback-Leibler divergence is laid out, the equation will favour mapping the points in the vicinity of the node, no matter their distance. Moreover, because the algorithm maps higher dimensional data to a lower dimensional, there's an inherent problem of crowding. To avoid these issues the tSNE algorithm uses the joint probability rather than the conditional probability, where the joint probability of the higher dimensional data is defined as

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{9}$$

.

As for the joint probability in the lower dimensional representation, rather than a Gaussian distribution, tSNE uses a one degree of freedom Student t-Distribution (Cauchy).

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_i - y_j\|^2)^{-1}} \tag{10}$$

.

Because of the heavier tail, it spreads the points outwards, therefore avoiding the cluttering of points that should be further apart. These newly defined joint probability also lessen the cost of the optimization.

Since the original paper, gives no details about the setting of the perplexity; Several different values were tried, as different

Fig. 3. Users press the 'Generate' button to get an image at random, next they select the type of clothes they want recommended and whether they want them to be male, female or unisex. Pressing the 'Recommend' button produces two columns one of which is chosen at random and the other with the help of our algorithm, the user then picks the column which better suits the formerly generated image

perplexities can have vastly different results, each showing a different aspect of the global or local structure of the data. [7]. We ran the algorithm for five thousand iterations after which we plotted the results. As is indicated in the paper, items of the same category cluster together, however while in the original paper, the different clusters were tightly woven with each other, in our results, while there is considerable overlap between categories, some categories formed groups of their own and were rather separated from each other. Whether this is simply because of the data or just an negligible intricacy of the tSNE algorithm is unclear. Though distances in tSNE may not have much meaning and in reality could only be a moderate distance away. [7] [5] While the tSNE algorithm did not converge for any of these values, this result persisted for different perplexity values ($Perp = 2, 10, 30, 50, 100, 200$). The tSNE visualization can be seen in (fig 4).

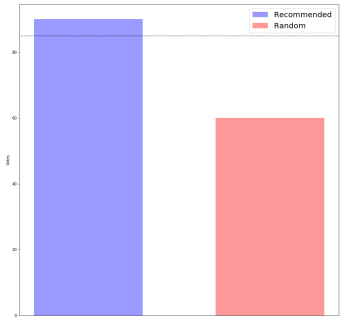

Fig. 4. Results from the user study. The dotted line indicates the cutoff for the statistically significant results (Obtained using a one tailed binomial distribution test, 95% confidence).

V. User Study

In order to test the effectiveness of the algorithm in [6] we conduct a user study. Users were given an image picked at random, next they were asked to select the type of garments they wished recommended to them based on the previously generated image. Two columns were generated, one of which was chosen at random and the other with the help of our algorithm, the surveyed users then picked the column which they thought better suited the formerly generated image. Therefore users couldn't distinguish between picking from the random column or from the algorithm one. We finally collected and compared the results of our survey.

During our survey, some users found several category names to be mistranslated, this could have affected their choice. Confusion could also have been averted by making the user interface more intuitive. Another problem was that due to the miscategorized items, our framework used the results from the third, fourth or fifth cluster rather than the first one as suggested in the paper. We suspect that we would have obtained better results were we to use the latter. Even so, our survey shows promising results. With 90 votes for the recommended items opposite the 60 votes for the random column, this result is statistically significant with a 95% confidence level (fig. 2).

VI. Conclusion

This paper describes an implementation of the concepts and framework first introduced in [6]. Using their pretrained Siamese network we were able to get the descriptors of our items. We built a database and a pipeline for clustering and recommending suitable pairs of clothes. We conducted a user study in which we asked users to rate whether or not the recommended items were suitable or not, thus testing the

performances of this algorithm. Finally, we visualized our data using the tSNE technique and noticed a difference with the plot in the original work. Further work can investigate the difference between our visualization and the one in the original paper, as well as focus on better fine-tuning the network or extending this approach to other fields like music, shopping and etc.

REFERENCES

[1] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. Graph.*, 34(4):98:1–98:10, July 2015.

[2] Ruining He and Julian McAuley. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, pages 507–517, Montréal, Québec, Canada, 2016. ACM Press.

[3] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based Recommendations on Styles and Substitutes. *arXiv:1506.04757 [cs]*, June 2015. arXiv: 1506.04757.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[5] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. 2008.

[6] Andreas Veit*, Balazs Kovacs*, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015. *The first two authors contributed equally.

[7] Martin Wattenberg, Fernanda Vigas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.