CONTENT2VEC: SPECIALIZING JOINT REPRESENTATIONS OF PRODUCT IMAGES AND TEXT FOR THE TASK OF PRODUCT RECOMMENDATION

Thomas Nedelec, Elena Smirnova & Flavian Vasile
Criteo Research
Paris, 32 Blanche, France
{t.nedelec,e.smirnova,f.vasile}@criteo.com

ABSTRACT

We propose a unified product embedded representation that is optimized for the task of retrieval-based product recommendation. We generate this representation using Content2Vec, a new deep architecture that merges product content information such as text and image and we analyze its performance on hard recommendation setups such as cold-start and cross-category recommendations. In the case of a normal recommendation regime where collaborative information signal is available we merge the product co-occurence information and propose a second architecture Content2vec+ and show its lift in performance versus non-hybrid approaches.

1 INTRODUCTION

Online product recommendation is now a key driver of demand, not only in E-commerce businesses that recommend physical products, such as Amazon (Marshall, 2006), TaoBao (Xiang, 2013) and Ebay (Academy, 2013), but also in online websites that recommend digital content such as news (Yahoo! (Agarwal et al., 2013), Google (Liu et al., 2010)), movies (Netflix (Bell & Koren, 2007)), music (Spotify (Johnson, 2015)), videos (YouTube (Covington et al., 2016)) and games (Xbox (Koenigstein et al., 2012)).

Two of the most challenging aspects of recommendation in general and of product recommendation in particular, are scalability and freshness. The first one addresses the problem of making fast recommendations, the second addresses the problem of updating recommendations based on real-time user interaction. One of the most encountered architecture solutions for recommendation at scale divides the recommendation process in two stages: *a candidate generation stage* that prunes the number of recommendable items from billions to a couple of hundreds, followed by a second *item selection stage* that decides the final set of items to be displayed to the user, as shown in Figure 1 (see (Mazare, 2016), (Cheng et al., 2016), (Covington et al., 2016)).

The first stage generally implies the pre-generation of an inverted-index over the set of recommendable products, paired with a real-time retrieval module, similarly to a search engine architecture. In our current paper we focus on the cases where the system supports vectorial product queries. The sources of the vectorial representations range from the set of co-occurring products, like in the case of neighborhood-based collaborative filtering, to a low-dimensional representation produced via matrix factorization or to an embedded representation produced via a deep neural network.

The second stage takes the candidate set and decides the final list of recommendations, usually by optimizing a ranking metric. This stage has in general a lot more constraints in terms of latency, due to its use of real-time signal that makes its predictions not cacheable. Therefore, in terms of model choice, the first stage can be a lot more complex than the second. In terms of impact, the quality of the candidate set coming from the first stage is crucial, since this constitutes a hard threshold on the performance of the second stage and of the overall system.

Because of the feasibility of using a more complex model and the potential impact on the final recommendation performance, we choose to concentrate our efforts on the task of optimal candi-

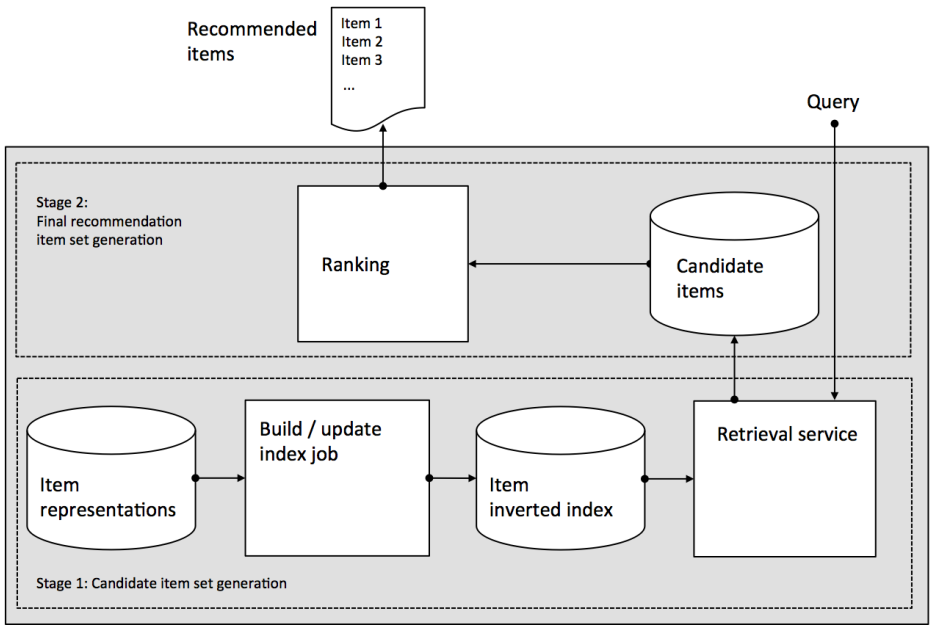

Figure 1: 2-Stage Recommender System Architecture.

date generation. We formalize the problem as a link prediction task, where given a set of past co-purchased products we try to predict unseen pairs of products. Related work in representation learning for recommendation investigated the use of collaborative filtering (CF), text and product images, but to our knowledge, there has been no attempt to unify all of these signals in a single representation. We see this as an opportunity to investigate the leveraging effect of generating a *Unified Product Representation* via a deep-learning approach. In the following, we formally define the set of associated requirements we would like to satisfy:

- **Relevance**: the representation should be optimized for product recommendation relevance, as measured by the associated target metrics (in this case, modeling it as a link prediction task and optimizing for the AUC of product pair prediction).
- **Coverage**: the representation should leverage all available product information (in our case, all product information available in the product catalog together with observed product co-occurrences).
- **Cross-modality expressiveness**: the representation should be able to account for interactions between various information sources such as text and image (can take into account the fact that the word ”red” and the ”red” color detector are correlated).
- **Pair-wise expressiveness**: the representation should be able to account for all interactions between pairs of products.
- **Robustness**: the representation should operate well (recommendation performance will not degrade dramatically) in hard recommendation situations such as product cold-start (new products, new product pairs) and cross-category recommendation. These are important use-cases in product recommendation, when the product catalog has high churn (as in the case of flash sales websites or classifieds) or the recommendation needs to leverage cross-advertiser signal (as in the case of new users and user acquisition advertising campaigns). This is a different goal from simply trying to optimize for relevance metrics, due to the inherent limitations of offline metrics in predicting future online performance.
- **Retrieval-optimized**: the representation should be adapted to a content-retrieval setup, both on the query and on the indexing side, meaning that the vectors should be either small, sparse or both.

We propose a modular deep architecture that leverages state-of-the-art architectures for generating embedded representations for image, text and CF input, re-specializes the resulting product embeddings and combines them into a single product vector. This is a very general architecture that can plugin any networks in the image and text domain and re-use them for the problem of product recommendation, along with their gains in representation learning for the two domains. We investigate multiple ways of merging the modality-specific product information and propose a new type of residual-inspired unit, which we name *Pairwise Residual Unit* that can model the joint aspects of the different product embeddings and show that it leads to good improvements.

We analyze our proposed architecture on an Amazon dataset (McAuley et al., 2015) containing information on co-purchased products. We report our improvements versus a text and an image-based baseline, that was introduced in previous work by McAuley et al. (2015) and show improvements both on normal and hard recommendation regimes such as cold-start and out-of-sample setups.

Our approach is similar with the recent work by Covington et al. (2016), that proposes a solution for video recommendation at YouTube within a similar architecture. Unlike their proposed solution, where, in order to support user vector queries, the candidate generation step co-embeds users and items, we are interested to co-embed just the product pairs, which generally have a much smaller dimension. In our approach, the personalization step can happen after the per-item candidates are retrieved.

Our main contributions are the following:

- We propose a novel way of integrating an item embedding model in the context of large scale recommender system with a 2-stage serving architecture and introduce the new task of *Unified Product Representation* for optimal candidate selection.
- We introduce a new deep architecture that merges content and CF signal for the task of product recommendation and propose the *Pairwise Residual Unit*, a new component that models the joint product representations.
- We introduce two novel experimental setups (hard cold start, out-of-category) and test that the proposed Content2Vec architecture satisfies the requirements we defined.

Though the focus of our work is on improving product recommendation through representation learning, we believe that simple extensions of our work can be applied to many other recommendation scenarios.

The rest of paper goes as follows: In Section 2 we cover previous related work and the relationship with our method. In Section 3 we present the Content2Vec model, followed by a detailed description of the resulting architecture in Section 4. In Section 5 we present the experimental setup and go over our results on Section 6. In Section 7 we summarize our findings and conclude with future directions of research.

2 Related Work

Our work fits in the new wave of deep learning based recommendation solutions, that similarly to classical approaches can fall into 3 categories, namely collaborative filtering based, content based or hybrid approaches.

Several approaches use shallow neural networks to build better item representations based on the co-occurrence matrix. The Prod2Vec algorithm (see (Grbovic et al., 2015)) implements Word2Vec ((Mikolov et al., 2013a), (Shazeer et al., 2016)), an algorithm that is at origin a shallow neural language model, on sequences of product ids, to reach a low-dimensional representation of each product. Among other embedding solutions that use the item relationship graph are the more recent extensions to Word2Vec algorithm such as Glove (Pennington et al., 2014) and SWIVEL (Shazeer et al., 2016) and the graph embedding solutions proposed in Node2Vec (Grover & Leskovec, 2016) and SDNE (Wang et al., 2016).

Content-based methods recommend an item to a user based upon an item description and a user profile ((Pazzani & Billsus, 2007)). This idea was deeply investigated in the information retrieval literature: in the context of web search, DSSM (Huang et al., 2013) and its extensions (Shen et al., 2014)(C-DSSM) and (Shan et al., 2016) are some of the most successful methods that specialize

query and document text embedding in order to predict implicit feedback signal such as document click-through rate. In the context of product recommendation, in (McAuley et al., 2015) the authors feed a pre-trained CNN (CNN trained on the ImageNet dataset, which is an image classification task that is very different from the task of image-based product recommendation) with products images and use the last layer of the network as the product embedding. This representation is subsequently used to compute similarities between products. Similarly, the authors in (Van den Oord et al., 2013) use CNNs to compute similarities between songs.

The performance of Collaborative Filtering (CF) models is often higher than that of content-based ones but it suffers from the cold-start problem. To take advantage of the best of both worlds, hybrid models use both sources of information in order to make recommendations. One possible way to incorporate product information is using it as side information in the product sequence model, as proposed in Meta-Prod2Vec (Vasile et al., 2016), leading to better product embeddings for products with low signal (low number of co-occurrences). In this work we continue the investigation of using both types of signal, this time both at training and product recommendation time.

Specializing content representation to recommendation Yosinski et al. (2014), the authors show that the low layers of DNNs trained on different tasks are often similar and that good performance can be reached by fine-tuning a network previously trained on another task. In the case of recommendation systems, this fine tuning was implemented by (Veit et al., 2015), where the authors specialize a GoogLeNet to the task of predicting co-view events based on product pictures.

3 CONTENT2VEC MODEL

Our proposed approach takes the idea of specializing the input representations to the recommendation task and generalizes it for multi-modality inputs, in order to leverage all product information and in particular, product images and product description text.

The main criteria for the Content2Vec architecture is to allow us to easily plugin new sources of signal and to replace existing embedding solutions with new versions. Also, we want to separate product-level embeddings and pair-level embeddings, such that the network can generate product vectors that are readily indexable. As a result, the Content2Vec architecture has three types of modules, as shown in Figure 2:

- **Content-specific embedding modules** that take raw product information and generate the associated vectors. In this paper we cover embedding modules for text, image, categorical attributes and product co-occurrences (for an example, see Figure 3).
- **Overall product embedding modules** that merge all the product information into a unified product representation.
- **Pair embedding module** that merges the product-to-product interactions and computes the final prediction. In the case of retrieval-optimized product embeddings, this module becomes the inner-product between the two items and all interactions between them are to be approximated within the product-level embedding modules.

Content2Vec training follows the architecture, learning module-by-module: In the first stage, we initialize the content-specific modules with embeddings from proxy tasks (classification for image, language modeling for text) and re-specialize them to the task of product recommendation. For the specialization task, as mentioned in Section 1, we frame objective as a link-prediction task where, given an incomplete set of pairs of products that were purchased together, we try to predict which pairs of products from a hold-out set were truly purchased together. We describe the loss function in Section 3.1.
In the second stage, we stack the modality-specific embeddings generated in the first stage into a general product vector and learn an additional residual vector using the same learning objective as in the specialization step. This will be described in depth in Section 4.2.
Finally, in the third stage, given the updated product vectors from stage two, we learn the linear combination between the similarities of the product vectors and make the final prediction.

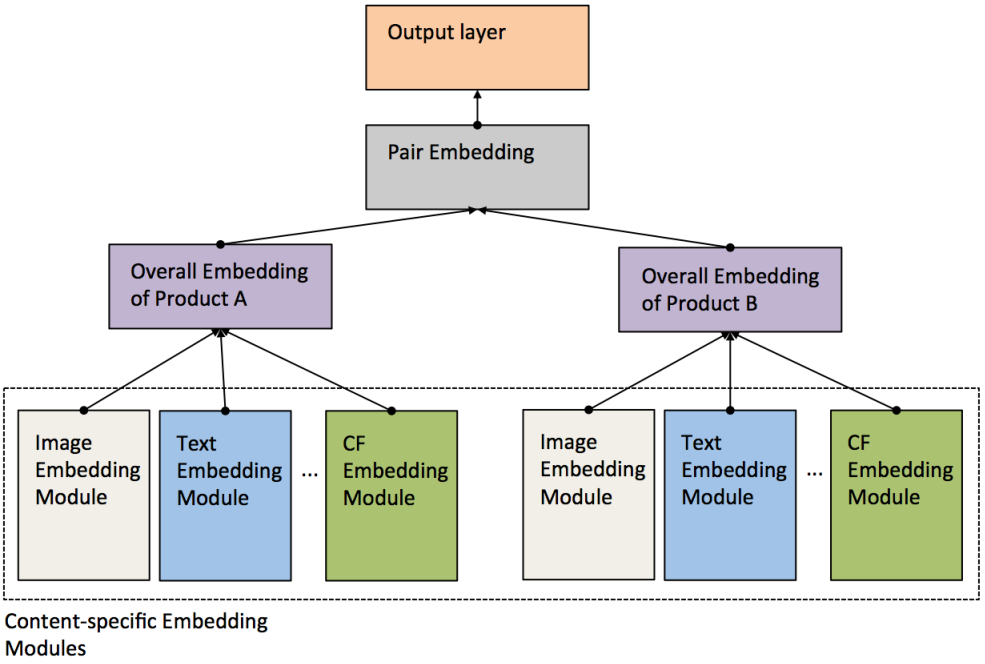

Figure 2: Content2Vec Architecture.

3.1 Loss Function

The previous work on learning pair-wise item distances concentrated on using ranking (McFee & Lanckriet, 2010), siamese (Hadsell et al., 2006) or logistic loss (Zheng et al., 2015). For optimizing the link prediction objective we choose the logistic similarity loss (eq. 1), that has the advantage of having a fast approximation via Negative Sampling loss (Mikolov et al., 2013b), shown in eq. 2. By using Negative Sampling, the prediction step can scale up to large number of items, by using all positive pairs and sampling the negatives on the fly.

$$L(\theta) = \sum_{ij} -X_{ij}^{POS} \log \sigma(sim(a_i, b_j)) - X_{ij}^{NEG} \log \sigma(-sim(a_i, b_j)) \tag{1}$$

$$L_{NS}(\theta) = \sum_{ij} -X_{ij}^{POS} (\log \sigma(sim(a_i, b_j)) + \sum_{l=1}^{k} \mathbb{E}_{n_l \sim P_D} \log \sigma(-sim(a_i, n_l)) \tag{2}$$

where:
$\theta = (a_i, b_j)$ is the set of model parameters, where a_i and b_j are the embedding vectors for the products A and B.
$sim(a_i, b_j) = \alpha < a_i, b_j > +\beta$ is the similarity function between a_i and b_j.
X_{ij}^{POS} is the frequency of the observed item pair ij (e.g. the frequency of the positive pair ij)
$X_{ij}^{NEG} = X_i - X_{ij}^{POS}$ is the frequency of the unobserved item pair ij (we assume that all unobserved pairs are negatives)
P_D probability distribution used to sample negative context examples n_l
k is a hyper parameter specifying the number of negative examples per positive example

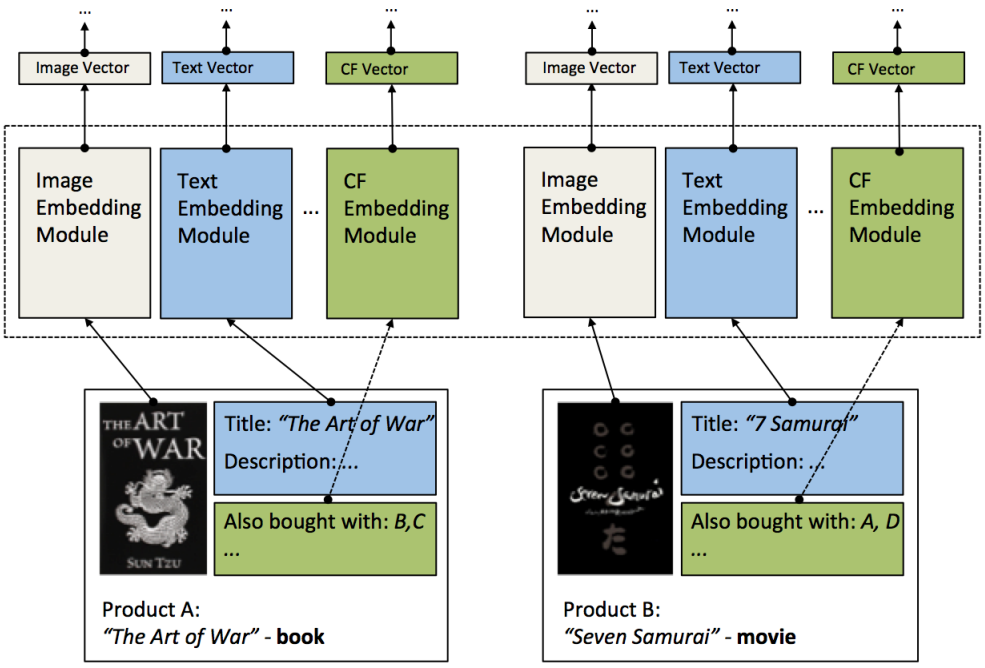

Figure 3: An example of using the content-specific modules to create embedded representations of two products with images, text and CF signal.

4 CONTENT2VEC MODULES

4.1 CONTENT-SPECIFIC EMBEDDING MODULES

Content-specific modules can have various architectures and are meant to be used separately in order to increase modularity. Their role is to map all types of item signal into embedded representations. In Figure 3 we give an illustrative example of mapping a pair of products to their vectorial representations.

In the following we analyze four types of input signal and embedding solutions for each one of them.

4.1.1 EMBEDDING PRODUCT IMAGES: ALEXNET

Model and proxy task: CNNs for Image Classification For generating the image embeddings we propose reusing a model trained for image classification, as in previous work by () and (He & McAuley, 2015). In (He & McAuley, 2015), the authors have shown how to use the Inception DNN architecture (Szegedy et al., 2015) and specialize it for the product recommendation task. However, the Inception architecture is very deep and requires extensive training time. For ease of experimentation we use AlexNet (), which is a simpler architecture that was also a winner on the ImageNet task (Krizhevsky et al., 2012) previously to Inception NN. In section 6 we will show that, even if simpler, when combined with additional product text information, the AlexNet-based solution can perform very well on the recommendation task.
For our experiments, we use the pretrained version of AlexNet available on Toronto's university website. We experimented with two different ways to specialize the representation in order to compute product similarities. In the first one, we learn a weighted inner product between the two representation (fc7 layer of ImageNet). In the second one, we back-propagate the error until the fc7 layer in order to specialize the layer to detect product similarities. The second approach lead to much better performance and is the one for which we report final results.

4.1.2 EMBEDDING PRODUCT TEXT: WORD2VEC AND CNNS ON SENTENCES

Model and proxy task: Word2Vec for Product Language Modeling For generating text embeddings, we propose reusing Word2Vec, a model for generating language models that has been employed in a various of text understanding tasks such as analogy detection (), sentiment analysis (), text classification (), translation () leading to state-of-the-art results while being conceptually simple. More recently, it has been shown in (Pennington et al., 2014) that Word2Vec is closely linked with matrix factorization techniques applied on the word co-occurence matrix. For Content2Vec, we chose to pretrain Word2Vec on the entire product catalog text information and not use an available set of word embeddings such as the one created on the Google Corpus (). The main reason is that the text distribution within product descriptions is quite different from the general distribution. For example the word *'jersey'* has a very different conditional distribution within the product description corpus versus general online text.
Text CNNs (Kim, 2014) offer a simple solution for sentence-level embeddings using convolutions. The convolutions act as a form of n-gram filters, allowing the network to embed sentence-level information and specializing word embeddings to higher-order tasks such as text classification or sentiment analysis. To the best of our knowledge, this is the first attempt to employ them for the task of product recommendation. For our task, we generate sentences based on the product titles and descriptions.

4.1.3 EMBEDDING PRODUCT CO-OCCURRENCES: PROD2VEC

Prod2Vec (Grbovic et al., 2015) is an extension of the Word2Vec algorithm to product shopping sequences. As a result, Prod2Vec can be seen as a matrix factorization technique on the product co-occurence matrix. In Content2Vec, the Prod2Vec-based similarity contains all of the information that can be derived simply from the sequential aspect of the user behavior, without taking into account the per-product meta-data.

4.1.4 EMBEDDING CATEGORICAL PRODUCT META-DATA: META-PROD2VEC

Meta-Prod2Vec (Vasile et al., 2016) improves upon Prod2Vec by using the product meta-data side information to regularize the final product embeddings. In Content2Vec, we can use the similar technique of co-embedding product categorical information with product ids to generate the embedding values for the categorical features. We did not use category embeddings in our experiments, because the training datasets contain only pairs from the same category.

4.2 JOINT PRODUCT EMBEDDING: PAIRWISE RESIDUAL UNIT

As stated in Section 1, the function of the product embedding module is two-fold: one, to model all interactions that exist between the modality-specific embeddings with respect to the final optimization objective, and second, to approximate the interaction terms between the products. With this in mind, we introduce a new type of learning unit, the *Pairwise Residual Unit* (eq. 4), which similarly to the original *residual unit* (eq. 3), allows the layers to learn incremental, i.e. residual representations (see Figure 4). In Section 6 we compare its performance (see *Content2Vec-res*, with similarity eq. 6), against the linear combination of the modality-specific similarities (denoted by *Content2Vec-linear*, with similarity eq. 5) and two models that take into model explicitly pair interactions, which are described in detail in Section 4.3). We show that the proposed architecture surpasses the performance of the linear model and has similar performance with the other two models, while allowing for a retrieval-based candidate scoring solution.

$$y = F(x) + x \tag{3}$$

$$y = SIM(F(x_1), F(x_2)) + SIM(x_1, x_2) \tag{4}$$

where:
x_1 and x_2 are the two product embedding vectors (obtained by stacking the modality-specific vectors)

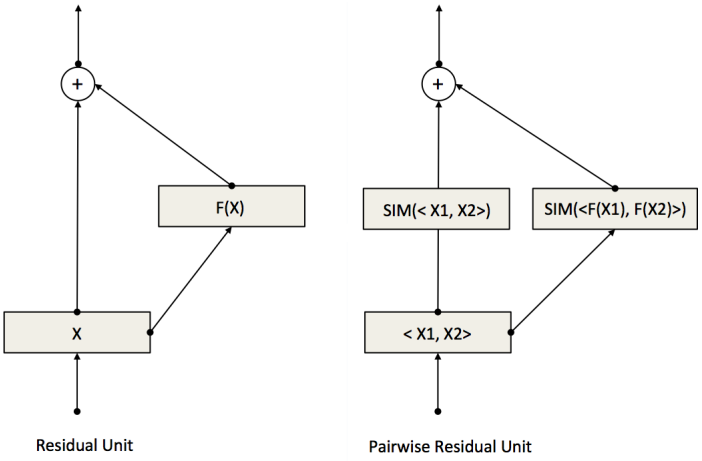

Figure 4: Pairwise Residual Unit

$SIM(.,.)$ is a similarity function over two embedding vectors x_1, x_2
$F(x)$ is a Rectified Linear Unit

$$SIM_{c2v}(a_i, b_j) = \sum_{m \in Modalities} w_m \sigma(SIM_m(a_i, b_j)) \tag{5}$$

$$SIM_{c2v-res}(a_i, b_j) = \sum_{m \in (Modalities + Residual)} w_m \sigma(SIM_m(a_i, b_j)) \tag{6}$$

4.3 Pair embedding module

In a retrieval-based architecture, the pair embedding module cannot support more than a simple linear combination of the product embedding vectors, such that the final score can be computed via innerproduct. However, we are still interested in knowing what is the trade-off in performance between an innerproduct-based candidate scoring and a model that allows for explicit interaction terms between the items. To this end, we introduce two explicit interaction models: *Content2Vec-crossfeat* - a model where we discretize the text and image-specific similarity scores and create explicit cross-features between them and *Content2Vec-embedpairs* - a model where we use a similar technique with *Paiwise Residual Unit*, in this case modeling the residual of the linear similarity directly as a vector in the pair embedding layer, as shown in Figure 5. In Section 6 we show that two models have as expected better performance than the linear model and that the non-linear embedding is slightly better.

5 Experimental Setup

Dataset We perform our evaluation on the publicly available Amazon dataset (McAuley et al., 2015) that represents a collection of products that were co-bought on the Amazon website. Each item has a rich description containing product image, text and category (any of the modalities can be missing). In terms of dimensionality, the dataset contains around 10M pairs of products. We concentrate on the subgraph of Book and Movie product pairs, because both categories are large and they have a reasonable sized intersection. This allows us to look at recommendation performance on cross-category pairs (to evaluate a model trained only on Book pairs on predicting Movie co-bought items) and mixed category pairs (to evaluate the models on Book-Movie product pairs).

Based on the full Book & Movies data we generate two datasets with different characteristics:

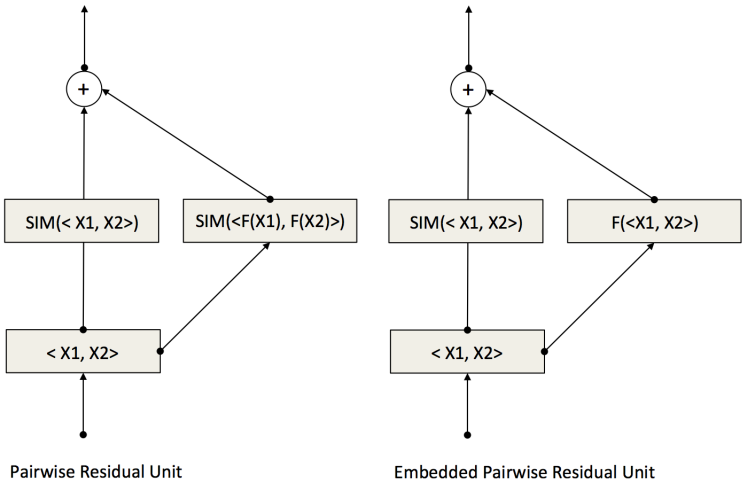

Figure 5: The two types of Pairwise Residual Units. By comparison with the first version that outputs a scalar, the second one outputs a vector that goes directly into the final prediction layer

The first dataset simulates a **hard cold start regime**, where all product pairs used in validation and testing are over products unseen in training. This tests the hardest recommendation setup, where all testing data is new. We decided to bench all of our hyperparameters on this regime and use the best setup on all datasets, since tuning on the harder dataset ensures the best generalization error.

The second dataset simulates a **non-cold start regime**, where the vast majority of the products in the test set are available at training time. The dataset is generated by taking the top 100k most connected products in the original dataset and keeping the links between them. This dataset tests for increase of performance of Content2Vec+ over a CF method on the most advantageous conditions for CF.

Evaluation task We evaluate the recommendation methods on the product link prediction task, similar to (He & McAuley, 2015). We consider the observed product pairs as positive examples and all unknown pairs as negatives. We generate negative pairs according to the popularity of the products in the positive pairs (negative examples between popular products are more likely to be generated) with a positive to negative ratio of 1:2.

Success metrics For the link prediction task, we use the Area Under Curve (AUC) of the Precision/Recall curve as our evaluation metric.

Competing methods

- *ImageCNN*: prediction based on specialized image embeddings similarity.
- *TextCNN*: prediction based on specialized text embeddings similarity.
- *Content2Vec-linear*: prediction based on the linear combination of text and image similarities.
- *Content2Vec-crossfeat*: prediction based on the linear combination of discretized image and text similarities and the crossfeatures between them.
- *Content2Vec-res*: prediction based on the linear combination of text and image similarities plus product-level residual vectors similarities.
- *Content2Vec-embedpairs*: prediction based on the linear combination of text and image similarities and a pair-level residual component.
- *Prod2Vec*: prediction based on the product vectors coming from the decomposition of the co-purchase matrix.

Recommendation Model	**Books**	**Movies**	**Mixed**
Models trained on Books dataset			
Book ImageCNN specialized	81%	78%	64%
Book TextCNN	72%	79%	76%
Book Content2Vec-linear	83%	83%	76%
Book Content2Vec-crossfeat	86%	83%	83%
Book Content2Vec-res	**89%**	83%	77%
Book Content2Vec-embedpairs	**90%**	82%	77%
Models trained on Movies dataset			
Movie ImageCNN specialized	59%	92%	60%
Movie TextCNN	63%	90%	65%
Movie Content2Vec-linear	64%	94%	65%
Movie Content2Vec-crossfeat	62%	94%	63%
Movie Content2Vec-res	60%	**95%**	66%
Movie Content2Vec-embedpairs	64%	**94%**	65%

Table 1: AUC results of Image and Text-based embeddings on Hard Cold Start dataset on Book, Movie and Mixed category test product pairs.

Cluster	Unigrams
Cluster 1:	mystic, qabalah, gnostic, qabalistic, kabbalah
Cluster 2:	puzzlers, backgammon, pathem, crossword, chess
Cluster 3:	mozilla, netscape, win32, perl, applescript
Cluster 4:	widowmaker, starfire, warhammer, drow
Cluster 5:	queueing, analogical, leibnitz, formalization, empiric
Cluster 6:	batman, kaiju, heavies, wolverine, joker
Cluster 7:	knot, sweater, placemats, pillowcase, napkin
Cluster 8:	furnishings, sotheby, junkmarket, house, furniture
Cluster 9:	beadwork, embroidery, quilt, bead, quilting
Cluster 10:	investing, diversify, stocks, markets, market

Table 2: Clusters of unigrams created by activating the trained TextCNN unigram convolutions.

- *Content2Vec+*: the method of predicting co-bought products based on the blend off all specialized embedding representations of each available product signal, namely ImageCNN, TextCNN and Prod2Vec.

6 RESULTS

In the following we analyze how *Content2Vec* satisfies the requirements outlined in Section 1. We have already shown through the architecture diagram in Figure 2 that *Content2Vec* can plug in easily any new product signal and defined embedding solutions for most popular product signal available for physical products. We go over the 5 other requirements and show experimentally that our proposed solution achieves good results on all of them.

Relevance The results in Table 1 show that the *Content2Vec* architectures (*Content2Vec-linear,crossfeat,res,embedpairs*) clearly outperform the models based on just one single type of signal (*ImageCNN* and *TextCNN*) on all evaluation scenarios (same category, different category, mixed).

Robustness In terms of robustness, the results on cross-category pairs show that model generalizes well and that the representations are sufficiently robust and improve even on out-of-sample data. As expected, the mixed category pairs proved to be hardest prediction task to improve. To further verify the universality of the resulting representations we ranked the word embeddings that activate the most the 100 unigram convolutions in the final TextCNN model. The results were very compelling, as show in in Table 2.

Product embedding expressivity In Table 1 we observe that the architecture choices that contain a product embedding layer (*Content2Vec-res* and *Content2Vec-embedpairs*) have the best performance on same-category evaluation, showing that we can model efficiently the interaction terms between modalities.

Pair embedding expressivity Adding pair-specific terms in *Content2Vec-crossfeat* and *Content2Vec-embedpairs* outperforms the simple baseline model *Content2Vec-linear* in the harder Books dataset and gives similar performance on Movies dataset.

Retrieval-ready The *Content2Vec-crossfeat* successfully approximates the pair-specific terms and shows better results than *Content2Vec-linear* in both datasets for the same category evaluation task.

7 CONCLUSIONS

This work has several key contributions. We show how to use all product signal for the task of product recommendation using a modular architecture that can leverage fast evolving solutions for each type of input modality. We define a set of requirements for evaluating the resulting product embeddings and show that our method leads to significant improvements over the single signal approaches on hard recommendation situations such as cold-start and out-of-sample evaluation. Finally, in order to model the join aspects of the product embeddings we introduce a new type of learning unit, named *Pairwise Residual Unit* and show the resulting gains on a real product co-purchases dataset.

8 NEXT STEPS

For the next steps, we want to pursue more aggressively for sparser and more compressed product representations, in order to help the performance of the final product retrieval system.

REFERENCES

DataStax Academy. Slideshare presentation. `http://www.slideshare.net/planetcassandra/e-bay-nyc`, March 2013. Accessed: 2016-04-08.

Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. Content recommendation on web portals. *Communications of the ACM*, 56(6):92–101, 2013.

Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. *arXiv preprint arXiv:1606.07792*, 2016.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 191–198. ACM, 2016.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1809–1818, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788627. URL `http://doi.acm.org/10.1145/2783258.2788627`.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. 2016.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1510.01784*, 2015.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 2333–2338. ACM, 2013.

Chris Johnson. algorithmic music recommendations at spotify, 2015.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Noam Koenigstein, Nir Nice, Ulrich Paquet, and Nir Schleyen. The xbox recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, pp. 281–284. ACM, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pp. 31–40. ACM, 2010.

Matt Marshall. Venture beat article. `http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/`, December 2006. Accessed: 2016-04-08.

PierreEmmanuel Mazare. Product recommendation at criteo. `http://labs.criteo.com/2016/09/product-recommendation-criteo/`, September 2016. Accessed: 2016-10-26.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52. ACM, 2015.

Brian McFee and Gert R Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 775–782, 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013b.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pp. 325–341. Springer, 2007.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D14-1162`.

Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. 2016.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. Swivel: Improving embeddings by noticing what's missing. *arXiv preprint arXiv:1602.02215*, 2016.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pp. 373–374. ACM, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pp. 2643–2651, 2013.

Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec-product embeddings using side-information for recommendation. *arXiv preprint arXiv:1607.07326*, 2016.

Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4642–4650, 2015.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. 2016.

Tracey Xiang. Technode article. `http://technode.com/2013/06/14/how-does-taobao-uses-user-data/`, June 2013. Accessed: 2016-04-08.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Lilei Zheng, Khalid Idrissi, Christophe Garcia, Stefan Duffner, and Atilla Baskurt. Logistic similarity metric learning for face verification. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1951–1955. IEEE, 2015.