

# Specializing Joint Representations for the task of Product Recommendation

Thomas Nedelec  
Criteo Research  
t.nedelec@criteo.com

Elena Smirnova  
Criteo Research  
e.smirnova@criteo.com

Flavian Vasile  
Criteo Research  
f.vasile@criteo.com

## ABSTRACT

We propose a unified product embedded representation that is optimized for the task of retrieval-based product recommendation. To this end, we introduce a new way to fuse modality-specific product embeddings into a joint product embedding, in order to leverage both product content information, such as textual descriptions and images, and product collaborative filtering signal. By introducing the fusion step at the very end of our architecture, we are able to train each modality separately, allowing us to keep a modular architecture that is preferable in real-world recommendation deployments. We analyze our performance on normal and hard recommendation setups such as cold-start and cross-category recommendations and achieve good performance on a large product shopping dataset.

## CCS CONCEPTS

•Computing methodologies →Machine learning; Neural networks;

## KEYWORDS

Recommender systems, representation learning, embeddings, second-order interactions

## ACM Reference format:

Thomas Nedelec, Elena Smirnova, and Flavian Vasile. 2017. Specializing Joint Representations for the task of Product Recommendation. In *Proceedings of DLRS 2017, Como, Italy, August 27, 2017*, 9 pages. DOI: 10.1145/3125486.3125489

## 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 [23], TaoBao [41] and Ebay [1], but also in online websites that recommend digital content such as news (Yahoo! - [2], Google - [22]), movies (Netflix - [4]), music (Spotify - [16]), videos (YouTube - [8]) and games (Xbox - [20]).

One of the most popular architectures for recommendation at scale (see [24], [7], [8]) divides the recommendation process in two stages: a *candidate generation stage* that prunes the number of recommendable items from a volume of potentially billions of items 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.(see Figure 3 in the Appendix for more details).

Due to lower constraints on the latency and the potential impact resulting from an improvement on the candidate generation stage, we choose to concentrate our efforts on the task of optimal candidate generation. We formalize the problem as a link prediction task, where given a set of past pairs of co-purchased products we try to predict the probability of being cobought for unseen pairs of products. Related work in representation learning for recommendation investigated the use of both collaborative filtering [9] and content information [25], but to our knowledge, there has been no attempt to unify them in a single representation. We see this as an opportunity to investigate the leveraging effect of generating a *Specialized Joint Representation* via a deep-learning approach.

In order to achieve this, we propose Content2Vec - a modular deep architecture that leverages state-of-the-art architectures for generating embedded representations for image, text and collaborative filtering (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 neural networks for modeling the input information and re-use them for the problem of product recommendation.

We argue that a modular architecture coupled with a module-by-module training is the easiest way to put such a complex model in production. In Content2Vec, most of the computation is spent on modeling the modality-specific representations, which in the case of periodic retraining should be able to leverage the previous models as the initialization states and converge very fast on the new input data. Furthermore, using pre-trained models for each modality allows us to leverage external sources of data and do transfer learning, whose value was repeatedly confirmed([42],[39]).

In the following, we formally define the set of associated requirements that define an optimal product embedding:

- **Relevance:** the representation should be optimized for product recommendation relevance, as measured by the associated target metrics (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).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DLRS 2017, Como, Italy

© 2017 ACM. 978-1-4503-5353-3/17/08...\$15.00  
DOI: 10.1145/3125486.3125489

- **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 analyze the performance of our proposed architecture on the five requirements presented above on an Amazon dataset [25] that contain information on co-purchased products. We report our improvements versus text, image ([25]) and collaborative-filtering ([9]) based baselines - and a simple combination of the three. We show improvements both on normal and hard recommendation regimes such as cold-start and cross-category setups.

Our main contributions are the following:

- We propose a novel way of integrating deep-learning item representation in the context of large scale recommender system with a two-stage serving architecture and introduce the new task of *Specialized Joint Representation* for optimal candidate selection in both cold start and normal recommendation setups.
- We introduce a new deep architecture that merges content and collaborative filtering signal for the task of product recommendation and propose the *Cross Interaction Unit* (CIU), a new learning component that models the joint product representation. We benchmark the different architectures in two experimental setups (hard cold start, cross-category) that test the robustness of our architecture.

The rest of the paper goes as follows: In Section 2, we cover previous related work and the relationship with our method. In Section 3, we present an overview of our new architecture and how we learn to compute similarities between products.

Section 4 details the Content-specific embedding modules that are used to build several representations from the different modalities: text, image and collaborative filtering data. In Section 5, we propose several architectures to fuse the modality-specific signals and build a unified product embedding. In Section 6, we present the experimental setup and go over the results in Section 6.2. 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

tree categories, namely collaborative filtering based, content based or hybrid approaches.

### 2.1 Collaborative filtering methods

Several approaches use neural networks to build better item representations based on the co-occurrence matrix. The Prod2Vec algorithm [9] implements Word2Vec ([27], [34]), 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 [31], SWIVEL [34], the graph embedding solutions proposed in Node2Vec [10] and SDNE [40].

### 2.2 Content based methods

Content-based methods recommend an item to a user based upon an item description and a user profile ([30]). This idea was deeply investigated in the information retrieval literature: in the context of web search, DSSM [14] and its extensions [35](C-DSSM) and [33] are some of the 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, [25] feed a pre-trained CNN with products images. The network was pretrained on the ImageNet dataset, an image classification task that is very different from the task of image-based product recommendation. The last layer of the network is used as the product embedding. This representation is subsequently used to compute similarities between products. Similarly, the authors in [37] use CNNs to compute similarities between songs. [42] 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 in [39], where the authors specialize a GoogLeNet architecture for predicting cobought events based on product pictures.

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 [38], 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.

### 2.3 Modeling Second Order Interactions

Modeling second order interactions is a key problem in Machine Learning since the introduction of the kernel trick by [5]. Recently, real world applications have approximated polynomial kernels by explicit cross features as shown in [6]. However, this approach can not scale to a very large number of features. Recent work proposed as a solution to factorize the second order terms and introduce Factorization Machines [32] and Field Aware Factorization Machines [17] that have achieved state of the art performance in many predictions tasks. Within the deep learning community, [33] managed to

model second order interactions by merging information through ReLUs. In our paper, we propose the Cross Interaction Unit, a simpler solution that allows fast convergence and good performance with modeling second order interactions.

In terms of architecture, our work is also similar to the one proposed by [8], that introduces a scalable solution for video recommendation at YouTube. 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 because for most ecommerce website the number of products is smaller than the number of website users. In our approach, the personalization step can happen after the per-item candidates are retrieved.

### 3 PROPOSED APPROACH: OVERVIEW

#### 3.1 Architecture

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

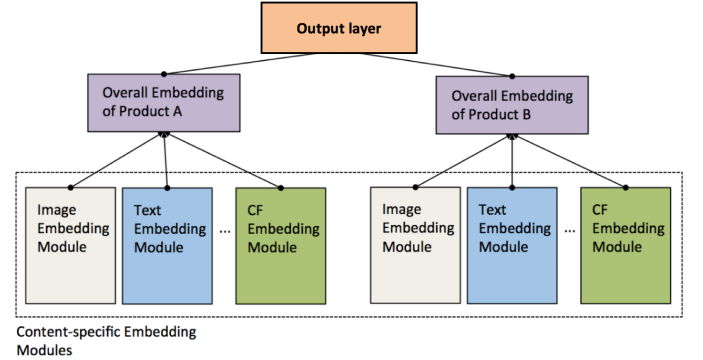
The main criteria for the architecture is to allow for the simple plugin of new sources of signal and for the upgrade of existing embedding solutions with new versions (e.g. to replace AlexNet with Inception NN for image processing). As a result, the Content2Vec architecture has three types of modules, as shown in Figure 1:

- **Content-specific embedding modules** that take raw product information and generate the product vectors. In this paper we cover embedding modules for text, image, categorical attributes and product co-occurrences (description of the different tested modules in Section 4).
- **The Joint Product Embedding modules** that merge all the product information into a joint product representation. The two different architectures for this module are detailed in Section 5.
- **The Output layer** that computes the probability for two products to be cobought or not (this layer is a sigmoid over the inner product between the two unified product embedding vectors)

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 the objective as a link prediction task where we try to predict the pairs of products purchased together. We describe the loss function in Section 3.2 and the different modules in Section 4.

In the second stage, we concatenate the modality-specific embedding vectors generated in the first stage into a general product vector that is fused into a joint representation using the second module. This will be described in depth in Section 5.

Finally, in the third stage, given the updated product vectors from stage two, we compute the final probability of being cobought using the output layer.



**Figure 1: Content2Vec architecture combines content-specific modules to produce embedding vector for each product, then uses these vectors to compute similarities between products. The modality-specific modules are presented in section 4 and the Joint Product Embedding module in Section 5**

#### 3.2 Learning a pair-wise item distance

We aim at learning a distance between products that is aligned with the probability of two products being of interest for the same user. The previous work on learning pair-wise item distances concentrated on using ranking loss [26] or siamese networks with L2 loss [11]. In [43], they introduce the logistic similarity loss :

$$L(\theta) = \sum_{ij} -X_{ij}^+ \log \sigma(\text{sim}(a_i, b_j)) - X_{ij}^- \log \sigma(-\text{sim}(a_i, b_j)) \quad (1)$$

where:

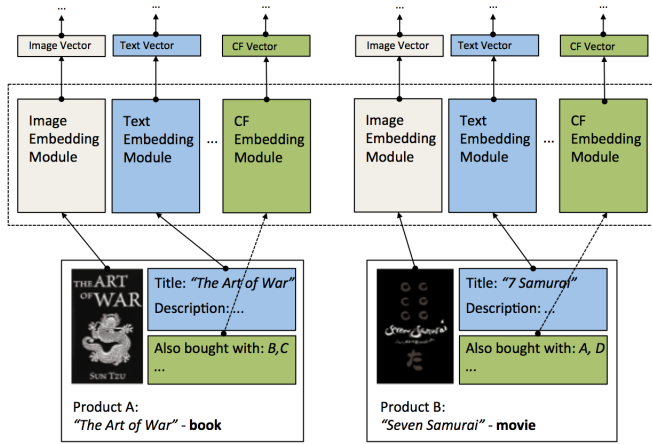
$\theta = (a_i, b_j)$  is the set of model parameters, where  $a_i$  and  $b_j$  are the embedding vectors for the products A and B,  $X_{ij}^+$  is the frequency of the observed item pair  $ij$  (e.g. the frequency of the positive pair  $ij$ ),  $X_{ij}^-$  is the frequency of the unobserved item pair  $ij$  (we assume that all unobserved pairs are negatives),  $\sigma$  is the sigmoid function and the similarity distance is defined as:

$$\text{sim}(a_i, b_j) = \alpha < a_i, b_j > + \beta \quad (2)$$

In the following, we detail the different modules used to learn the distance between products. Based on these modules, we can compute some similarities between products based either on their text, their image or their collaborative filtering data. We combine these metrics in the final module. These modules could also be used on their own since they are trained separately to predict whether two products are related or not.

### 4 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 2 we give an illustrative example of mapping a



**Figure 2: An example of using the content-specific modules to create embedded representations of two products with images, text and CF signal.**

pair of products to their vector representations. In the following we analyze four types of input signal and embedding solutions for each one of them.

#### 4.1 Embedding product images with AlexNet

For generating the image embeddings we propose reusing a model trained for image classification, as in previous work by [21] and [13]. In [13], the authors have shown how to use the Inception architecture [36] 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 [21] prior to Inception NN or more recently to ResNet [12]. In section 6.2 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. Our architecture can be seen as a siamese network: the same network is used to build the representation of product A and product B based on their respective image. We specialize the fc7 layer (last layer of AlexNet) to detect product similarities by minimizing the negative sampling loss presented in the previous section. In eq.1,  $a_i$  is the value of the fc7 layer when the input is the image of product A and  $b_j$  the same for product B. We only minimize the loss with respect to the weights between fc6 and fc7 since [42] proved that it was sufficient to reach good performance (and save a lot of computational resources). At the end of the optimisation process, we reach visual based embeddings that are specialized to predict if two products could be co-bought or not.

#### 4.2 Embedding product text with Word2Vec and CNN on sentences

With this module, we want to compute products similarities based on their text descriptions. The module is trained in two steps. First,

we train a Word2Vec [28] model in order to reach a representation of each word of the catalogue. Then, we train a TextCNN [18] that combines the words representation to build a product text embedding trained to detect whether two products were cobought or not. To the best of our knowledge, this is the first attempt to employ a TextCNN architecture for the task of product recommendation.

For generating word embeddings, we propose reusing Word2Vec, a model for generating language models that has been employed in a various of text understanding tasks. More recently, it has been shown in [31] that Word2Vec is closely linked with matrix factorization techniques applied on the word co-occurrence matrix. We chose to train 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.

TextCNN offers 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. We learn the filters by minimizing the negative sampling loss to predict if two products were cobought. The trainable parameters of the model are the weights in the convolutional layer that we call filters.

We keep fixed the word embeddings that were trained through Word2Vec as recommended in [18]. Using embeddings helps to generalize to words that were not frequent in the training set. The architecture has the advantage to allow to choose an arbitrary length for the text input. The two siamese networks are fed by the word embeddings of the first 10 token of the concatenated product title and description of product A and B. We only tested with keeping the first 10 tokens for ease of experiments (it is straightforward to extend to a higher number of considered tokens). We provide a comparison of performance with the image-based module in the experiments section.

#### 4.3 Embedding product co-occurrences with Prod2Vec

Prod2Vec [9] 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-occurrence matrix (see [31]). In Content2Vec, the Prod2Vec-based similarity contains all of the information that can be derived from the sequential aspect of the user behavior, without taking into account the per-product meta-data.

#### 4.4 Embedding categorical product meta-data with Meta-Prod2Vec

Meta-Prod2Vec [38] 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.

## 5 THE JOINT PRODUCT EMBEDDING MODULE

With these different modules, we can build several product embeddings that are modality-specific and specialized for the task of product recommendation. In the next stage, we combine them in order to build a unified representation of the product. This representation will consider all the signals available on the product: text, image and collaborative filtering data.

### 5.1 Joint Product Embedding with performance constraints

As stated in Section 1, the function of the product embedding module is two-fold: first, to model all interactions that exist between the modality-specific embeddings with respect to the final optimization objective, and second, to approximate interaction terms between the products that cannot be explained by a linear combination of the modality-specific similarities. With this in mind, we introduce a new type of learning unit, the *Cross Interaction Unit* (eq. 4).

We note  $X_1$  (respectively  $X_2$ ) the two product embedding vectors (obtained by stacking the modality-specific vectors):

$$X_1 = [X_1^{mod_1}, X_1^{mod_2}, \dots, X_1^{mod_N}] \quad (3)$$

where  $mod_i$  are the different modalities considered by the architecture. We define the *Cross Interaction Unit* as:

$$y = \text{sim}(F(X_1), F(X_2)) + \text{sim}(X_1, X_2) \quad (4)$$

where:

$\text{sim}(\cdot, \cdot)$  is a similarity function over two embedding vectors  $X_1, X_2$ ,  $F(x)$  is a ReLU layer.

The  $F$  function can be seen as a module that will use information coming from all the modalities to explain similarities that could not be explained by simply using their linear contribution.

To be able to measure the incremental value of introducing this residual vector we introduce a baseline architecture that computes the final prediction based on the linear combination of the modality-specific similarities denoted by *Content2Vec-linear* with the associated similarity function defined in eq. 5.

$$\text{sim}_{c2v-lin}(X_1, X_2) = \sum_{m \in \text{Modalities}} w_m \text{sim}_m(X_1^m, X_2^m) \quad (5)$$

Under this notation, the CIU-based architecture denoted as *Content2Vec perf* minimizes the logistic loss with the similarity function defined in eq. 6.

$$\text{sim}_{c2v-res}(X_1, X_2) = \sum_{m \in (\text{Modalities} + \text{Residual})} w_m \text{sim}_m(X_1^m, X_2^m) \quad (6)$$

In order to learn the residual vector, we keep fixed the modality-specific similarities and co-train the final weights of each of the modalities together with the product-specific residual layer. For example, in the case of using only image and text signals, our final predictor can be defined as in the following equation where  $P_{txt}$  and  $P_{img}$  are pre-set and  $w_{txt}$ ,  $w_{img}$ ,  $w_{res}$  and  $P_{res}$  are learned together:

$$P(\text{pos}|X_1, X_2) = \sigma \left( w_{txt} P_{txt}(\text{pos}|X_1^{txt}, X_2^{txt}) + w_{img} P_{img}(\text{pos}|X_1^{img}, X_2^{img}) + w_{res} P_{res}(\text{pos}|X_1^{res}, X_2^{res}) \right) \quad (7)$$

with:

$$P_{res}(\text{pos}|X_1, X_2) = \alpha < F([X_1^{txt}, X_1^{img}]), F([X_2^{txt}, X_2^{img}]) > + \beta \quad (8)$$

In Section 6.2 we compare the performance of *Content2Vec-perf* and *Content2Vec-linear* and show that, as expected, the proposed architecture surpasses the performance of the linear model, while allowing for a retrieval-based candidate scoring solution.

### 5.2 Joint Product Embedding with size constraints

One of the main objectives of the paper is to investigate architectures that can reach a unified product embedded representation that is optimized for the task of retrieval-based product recommendation. In order to enable an efficient retrieval task ([15] and [3]), we implement an architecture that places a constraint on the size of the final vector representing the product of small dimension (we set arbitrarily the allowed dimension to 200 in our case). To this end, we represent the product representation module by a fully connected ReLU layer that compresses the representation of the product into a vector of size 200. This layer takes as input a concatenated version of all the vectors available on the products from the modality-specific modules. After this compression layer, we reach one unified vector for product A and one unified vector for product B. These vectors are then used to compute product similarities by computing the inner product between two products representations. In the following, we will refer to this architecture by *Content2Vec-compressed*.

The product representations coming from the modality-specific modules are already relevant w.r.t the final task of product pair prediction, as shown by the performance of Image and TextCNN baselines in section 6.2. In order to avoid losing information coming from the concatenated vector representation, different initializations of the fully connected layer are possible. Of course, the initialization that would keep the incoming information from the trained modalities networks would be a layer close to the identity matrix. One way to approximate this objective would be to compute the PCA over vector representation of the full training dataset, but this is computationally expensive. We propose a simpler initialization that considers 200 random products (the number of products considered is defined by the dimension we want to impose to the final vector) from the training dataset and uses their representation as rows of the matrix. Hence, at the beginning, the layer will compute the similarities between each product and these 200 source products.

Model trained on Books dataset	Books	Movies	Mixed
<i>Modality Baselines</i>			
ImageCNN	81%	78%	64%
TextCNN	72%	79%	76%
<i>Fusion Baselines</i>			
Fusion-linear	83%	<b>83%</b>	76%
Fusion-crossfeat	86%	<b>83%</b>	<b>83%</b>
<i>Our approaches</i>			
Content2Vec-compressed	85%	81%	64%
Content2Vec-perf	<b>89%</b>	<b>83%</b>	77%
Model trained on Movies dataset	Movies	Books	Mixed
<i>Modality Baselines</i>			
ImageCNN	92%	59%	60%
TextCNN	90%	63%	65%
<i>Fusion Baselines</i>			
Fusion-linear	94%	<b>64%</b>	65%
Fusion-crossfeat	94%	62%	63%
<i>Our approaches</i>			
Content2Vec-compressed	<b>95%</b>	54%	58%
Content2Vec-perf	<b>95%</b>	60%	<b>66%</b>

**Table 1: Comparative results in terms of AUC between the different architectures on the hard cold start dataset (test set: same category as during training, different category as during training, mixed category)**

## 6 EXPERIMENTAL RESULTS

### 6.1 Dataset

We perform our evaluation on the publicly available Amazon dataset [25] 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. 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:

- 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 (results shown in Table 1).
- The second dataset simulates a **soft cold start regime**, where some of the products in the test set are available at training time. The dataset is generated by taking the top 200k most connected products in the original dataset and

sampling 10% of the links between them (results shown in Table 2).

*Hyper-parameters.* We fixed the sizes of embedding vectors for image CNN module to 4096 hidden units, for text CNN module to 256, for Prod2Vec module to 50, for residual representation to 128. For optimization, we use the Adam algorithm ([19]) and we manually set the initial learning rate based on the validation set performance. The batch sizes vary for different datasets. We train all the models until validation set performance stops increasing.

*Loss.* Instead of minimizing the logistic loss, we minimize the Negative Sampling loss [29] which is a fast approximation of the logistic loss. The prediction step can scale up to a large number of items, by using all positive pairs and sampling the negatives on the fly.

*Evaluation task.* We evaluate the recommendation methods on the product link prediction task, similar to [13]. We consider the observed product pairs as positive examples and all unknown pairs as negatives. We generate negative pairs according to the frequency 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.

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

*Baselines.* We introduce several baselines and compare their performances with our proposed architectures:

Recommendation Models	Test
<i>Baselines</i>	
ImageCNN	80%
TextCNN	78%
Prod2vec	86%
Fusion-linear	88%
Fusion-linear+	89%
<i>Without Prod2vec signal</i>	
Content2vec-compressed	87%
Content2vec-perf	89%
<i>With Prod2vec signal</i>	
Content2vec-compressed+	89%
Content2vec-perf+	92%

**Table 2: Comparative results in terms of Area Under Precision-Recall Curve (AUPRC) between the different architectures on the soft cold start dataset**

- *ImageCNN*: prediction based on specialized image embeddings similarity (that previously showed state-of-the-art results on the Amazon dataset [25])
- *TextCNN*: prediction based on specialized text embeddings similarity
- *Prod2Vec*: prediction based on the product vectors coming from the decomposition of the co-purchase matrix
- *Fusion-Linear*: prediction based on the linear combination of text and image similarities
- *Fusion-Crossfeat*: prediction based on the linear combination of discretized image and text similarities and their conjunctions: we bucketize the text and image-specific similarity scores and create explicit feature conjunctions between them.

*Our approaches.*

- *Content2Vec-compressed*: prediction based on the compressed version of the product representation
- *Content2Vec-perf*: prediction based on the linear combination of text and image similarities plus product-level residual vectors similarities
- *Content2Vec+*: prediction based on the ensemble of Prod2Vec and Content2Vec models

## 6.2 Results

The two following tables (Tables 1, 2) correspond to the two types of dataset we consider: the hard cold start dataset where all product pairs used in validation and testing are over products unseen in training and the soft one where some of the products in the test set are available at training time.

**6.2.1 Difference of performance between baselines.** To the best of our knowledge, no study has been made so far on the performance of TextCNN in the recommendation system setting. We observe that it is slightly worse than ImageCNN when evaluated on the same product category it was trained. On the mixed dataset (pairs of products from both Books and Movies), TextCNN generalize better than ImageCNN. We cannot bench Prod2Vec on the hard

cold start dataset since no collaborative filtering data on the test products were available at training time. When good collaborative filtering data is available, Prod2Vec outperforms both TextCNN and ImageCNN but the performance of Prod2Vec strongly depends on the degree of connectivity of the products graph. For new products, Prod2Vec can not be used.

**6.2.2 Combining product signals.** The results show that first our two main proposed method *Content2Vec-compressed* and *Content2Vec-perf* can leverage the additional signal provided by each of the input modalities in a joint manner and leads to significant gains in AUC versus the one-signal baselines (ImageCNN, TextCNN) and their linear combination (*Content2Vec-linear*).

From the point of view of robustness, *Content2Vec-perf* learns product representations that perform better than the baseline methods on out-of-sample recommendations such as cross-category pairs and mixed-category pairs (Table 1). However, there still exists a gap for improvement.

**6.2.3 Tradeoff between performance and reaching a compressed product representation.** *Content2Vec-compressed* performs slightly worse than *Content2Vec-perf* but it uses a more compressed representation than enables a more efficient retrieval system. We also remark that the distance and the representation learned by *Content2Vec-compressed* is more category-specific than *Content2Vec-perf* since *Content2Vec-perf* is performing better in the cross-category setting. Besides, the training time before reaching a good *Content2Vec-compressed* model is higher than *Content2Vec-perf*. Hence, there exists a clear trade-off in order to choose between *Content2Vec-perf* and *Content2Vec-compressed*: in terms of prediction performance, training time and cross-category efficiency, *Content2Vec-perf* has better results, while *Content2Vec-compressed* still achieves good prediction performance and offers a compressed product representation.

**6.2.4 Incorporating Prod2Vec signal.** *Content2Vec-perf+*, our proposed hybrid architecture that combines content and CF signal achieves better performance than the content and CF-only models. It confirms that even on a setting where good collaborative data



are available, the representation can be improved by using all other signals. We find also interesting that our content-based method *Content2Vec-perf* has a similar performance than a collaborative-filtering based method such as *Prod2Vec*.

## 7 CONCLUSIONS

In this paper, we propose *Content2vec*, a new product representation architecture which addresses most of the requirements outlined in the *Joint Product Representation* task. It generates relevant representations by optimizing for the target offline metric i.e AUC of hold-out product pairs prediction and covers all input signal by using a representation module for each type of signal. It also offers cross-modality expressiveness by the introduction of the product embeddings modules and optionally pair-wise expressiveness in the pair embedding module, passes robustness checks by performing better than baselines on hard cold start and cross-category evaluation tasks, and offers the possibility for retrieval-optimized vectors with the *Content2vec-compressed* version.

This work has several key contributions: We develop a method that is able 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 cross-category evaluation. We show how to build a compressed product representation that is able to take into consideration all signal available on the product to perform well on some hard cold start setting and improve the collaborative filtering representation in a normal recommendation scenario. Finally, in order to model the joint aspects of the product embedding with keeping some linearities in the model we introduce a new type of learning unit, named *Cross Interaction Unit* and show the resulting gains on a real product co-purchases dataset.

For the next steps, we would like to improve *Content2vec* for cross-category tasks and impose some sparsity constraints on product representations to increase the performance of the final product retrieval system.

## REFERENCES

- [1] DataStax Academy. 2013. Slideshare presentation. <http://www.slideshare.net/planetcassandra/e-bay-nyc>. (2013). Accessed: 2016-04-08.
- [2] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. 2013. Content recommendation on web portals. *Commun. ACM* 56, 6 (2013), 92–101.
- [3] Artem Babenko, Relja Arandjelović, and Victor Lempitsky. 2016. Pairwise Quantization. *arXiv preprint arXiv:1606.01550* (2016).
- [4] Robert M Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter* 9, 2 (2007), 75–79.
- [5] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 144–152.
- [6] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2015), 61.
- [7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishu Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ipsir, and others. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
- [8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
- [9] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. 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)*. ACM, New York, NY, USA, 1809–1818. DOI: <http://dx.doi.org/10.1145/2783258.2788627>
- [10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.
- [11] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
- [13] Ruining He and Julian McAuley. 2015. VBPR: visual bayesian personalized ranking from implicit feedback. *CoRR* (2015).
- [14] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. 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*. ACM, 2333–2338.
- [15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061* (2016).
- [16] Chris Johnson. 2015. fiAlgorithmic Music Recommendations at Spotify. (2015).
- [17] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. 2017. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 680–688.
- [18] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
- [19] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [20] Noam Koenigstein, Nir Nice, Ulrich Paquet, and Nir Schleyen. 2012. The Xbox recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 281–284.
- [21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
- [22] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 31–40.
- [23] Matt Marshall. 2006. Venture Beat article. <http://venturebeat.com/2006/12/10/aggregate-knowledge-raises-5m-from-kleiner-on-a-roll/>. (Dec. 2006). Accessed: 2016-04-08.
- [24] P.E Mazare. 2016. Product Recommendation at Criteo. <http://labs.criteo.com/2016/09/product-recommendation-criteo/>. (2016).
- [25] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.
- [26] Brian McFee and Gert R Lanckriet. 2010. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 775–782.



- [27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR workshop* (2013).
- [28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
- [29] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26*, C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.). Curran Associates, Inc., 2265–2273. <http://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive-estimation.pdf>
- [30] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.
- [31] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. <http://www.aclweb.org/anthology/D14-1162>
- [32] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.
- [33] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 255–262.
- [34] Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving Embeddings by Noticing What’s Missing. *arXiv preprint arXiv:1602.02215* (2016).
- [35] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 373–374.
- [36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.
- [37] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.
- [38] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 225–232.
- [39] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. 2015. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*. 4642–4650.
- [40] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1225–1234.
- [41] Tracey Xiang. 2013. TechNode article. <http://technode.com/2013/06/14/how-does-taobao-uses-user-data/>. (June 2013). Accessed: 2016-04-08.
- [42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in neural information processing systems*. 3320–3328.
- [43] Lilei Zheng, Khalid Idrissi, Christophe Garcia, Stefan Duffner, and Atilla Baskurt. 2015. Logistic similarity metric learning for face verification. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1951–1955.

## A INFRASTRUCTURE OF THE RECOMMENDATION SYSTEM

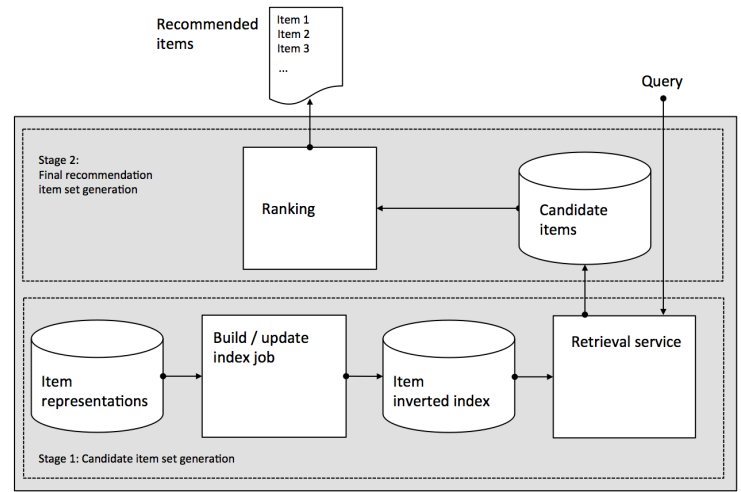


Figure 3: 2-Stage Recommender System Architecture.