



**HAL**  
open science

# Deep Multi-View Learning for Tire Recommendation

Thomas Ranvier, Kilian Bourhis, Khalid Benabdeslem, Bruno Canitia

► **To cite this version:**

Thomas Ranvier, Kilian Bourhis, Khalid Benabdeslem, Bruno Canitia. Deep Multi-View Learning for Tire Recommendation. 2021 International Joint Conference on Neural Networks (IJCNN), Jul 2021, Shenzhen, China. pp.1-8, 10.1109/IJCNN52387.2021.9534318 . hal-03614780

**HAL Id: hal-03614780**


**<https://hal.science/hal-03614780v1>**

Submitted on 22 Mar 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Deep Multi-View Learning for Tire Recommendation

Thomas Ranvier   
Université Lyon1 - LIRIS  
43 bd du 11 Novembre 1918  
69622 Villeurbanne, France  
ranvier.thomas.pro@gmail.com

Khalid Benabdeslem  
Université Lyon1 - LIRIS  
43 bd du 11 Novembre 1918  
69622 Villeurbanne, France  
khalid.benabdeslem@univ-lyon1.fr

Kilian Bourhis  
Lizeo IT  
42 Quai Rambaud  
69002 Lyon, France  
kilian.bourhis@lizeo-group.com

Bruno Canitia  
Lizeo IT  
42 Quai Rambaud  
69002 Lyon, France  
bruno.canitia@lizeo-group.com

**Abstract**—We are constantly using recommender systems, often without even noticing. They build a profile of our person in order to recommend the content we will most likely be interested in. The data representing the users, their interactions with the system or the products may come from different sources and be of a various nature. Our goal is to use a multi-view learning approach to improve our recommender system and improve its capacity to manage multi-view data. We propose a comparative study between several state-of-the-art multi-view models applied to our industrial data. Our study demonstrates the relevance of using multi-view learning within recommender systems.

**Index Terms**—multi-view, recommender system, attention mechanisms, deep learning

## I. INTRODUCTION

Many datasets contain data of various natures or coming from different sources, those are called multi-view data. Multiple views can provide additional information compared to a single view, intelligently exploiting the multi-view aspect of such data can lead to an improvement in learning performance. The main challenge of multi-view learning is not only to exploit the consensus knowledge contained in all views but also to exploit the complementarity of the views to improve performance.

In this paper we present a comparative study in a real-life application context on industrial data between a Baseline model that does not exploit the multi-view aspect of the data and three state-of-the-art models taking advantage of the multi-view aspect of the data. We provide several results of different models showing the advantage of multi-view learning. We thus demonstrate the relevance and interest of multi-view learning on a tire recommendation problem in an industrial context.

### A. Recommender systems

The purpose of a recommender system is to estimate the products that are most likely to be of interest to a user [1]. There exists two main recommendation tasks: the prediction of a rating and the creation of a ranking. The prediction of a

rating is aimed at estimating the score that the user could give to a product and is generally based on the scores given to other products by the same user. The point of creating a ranking is to define a ranked list composed of  $n$  products ordered by the user’s estimated taste preference. It is usually called a “top- $n$ ” [2]. There are three main recommender system categories [3]:

- 1) Content-based recommendations. The user is recommended products similar to those that he has previously consulted and appreciated [1]. Recommendations are built from the user profile, they are based on the similarity between the products the user is interested in. To decide which product to recommend the system calculates a utility value  $util(u, v)$  between the user  $u$  and an item  $v$  [3], usually defined as follows:

$$util(u, v) = score(profile(u), content(v)) \quad (1)$$

The system creates a profile for each user containing its tastes and preferences, noted  $profile(u)$ . It does the same for each product, noted  $content(v)$ . The two main drawbacks of this type of recommender system are the cold start and overspecialization problems. Indeed, it is hard to make pertinent recommendations to an unknown user, since its profile is empty, and conversely it is hard to properly recommend a new item that nobody has ever consulted or purchased, this is called the cold-start problem [4]. Overspecialization can also appear, this has the effect of enclosing the recommendations made to the user in a bubble and does not allow the recommendation of new and original products.

- 2) Collaborative Filtering. The user is recommended products that people with similar taste have liked before. In that case the utility value  $util(u, v)$  between the user  $u$  and the item  $v$  is estimated based on the utility values  $util(u_j, v)$  assigned  $v$  by users  $u_j \in U$  which are similar to  $u$  [3]. In this way, users with similar profiles will be recommended similar products. That type of

recommender system faces the cold-start problem too but is less subject to the overspecialization problem than content-based systems. It also faces the sparsity problem, since it is based on the rating of products by users and that each user only rates a very small part of available products.

- 3) Hybrid approaches. The point of hybrid approaches is to combine the two previously described approaches together. The main point of combining content-based recommendations and collaborative filtering is to avoid specific limitations presented by each of those approaches [5], [6]. The most common approach in recent years has been to design a model combining the two approaches [3], thus making it possible to solve, at least partially, the cold-start and sparsity problems [1].

A lesser known and used category is *session-based* recommender systems [7]. Those use the similarities between user's browsing habits and their relationship to the consulted products to produce a recommendation. In that case it is important to take into account the sequential nature of browsing sessions data. It is therefore needed to use a model that is able to handle a temporal dimension, such as an RNN (Recurrent Neural Network) [8] or a 3D-CNN (3D Convolutional Neural Network) [7].

### B. Multi-view learning

A lot of data is collected from distinct sources. Making intelligent use of these different views can improve the performance of a classification system. Multi-view learning can also be applied to improve performance on data organized in a single view by manually generating multiple views [9].

In order for multi-view learning to be effective it is necessary that the views adhere to certain principles, otherwise the use of multiple views could lead to performance degradation. The two main principles of multi-view learning are the consensus principle and the complementary principle [10]:

- 1) Consensus principle. In supervised learning, this principle defines that by minimizing disagreements between each view, the error rate on each view will also be minimized. In other words, a system capable of correlating the results obtained from different views will at the same time minimize classification errors on each of these views.
- 2) Complementary principle. This principle defines that in the context of multiple views, each view of the data contains knowledge that the others do not possess, thus, multiple views can be exploited to comprehensively and accurately describe the data and lead to better learning performance.

By exploiting these two principles, it is theoretically possible to ensure a learning performance improvement by comparison to single view learning.

### C. Multi-view recommender systems

A recommender system can benefit from multi-view learning provided that the used data allows the application of the

consensus and complementary principles. Concrete applications of the multi-view approach in the field of recommendation usually make use of deep learning composite models. Several state-of-the-art models have been considered to be applied to our industrial data and be part of our comparative study.

- Wide & Deep model [11]: This model combines a linear model with a deep model, allowing the benefits of each model to be exploited within a unified system. Deep models are capable of generalizing to unknown data but tend to overgeneralise when insufficient training data is used. Linear models are capable of storing exception rules on scattered data with a lower generalization capacity but a lower need for the number of parameters to be learned. The wide component of the model applies the linear model to a set of features, which can be raw or transformed with simple operations, such as vector products, and the deep component is a deep forward propagation neural network. Before being passed to the deep network, the scattered, high-dimensional data is converted into smaller, dense vectors, called embedding vectors. The model is trained as a whole using joint training, which allows multi-branch models to be trained using a single error function [12].
- DeepFM model [13]: This model is based on the observation that the Wide & Deep model requires extensive manual analysis work on the data in order to determine the best mathematical transformations to use for the wide component. The DeepFM model automates this manual work using a factorization machine (FM), which allows the modeling of pairwise interactions between data features using vector products between embedding vectors. The deep component of the model remains unchanged compared to that of the Wide & Deep model. The model is also driven as a single model using joint training.
- NeuMF model [14]: This model generates two latent vectors from the user's view and two others from a product view. The two vectors from each view are generated with matrix factorization (MF vector) and with a multi-layer perceptron (MLP vector). The NeuMF model shares similarities with the DeepFM, both models were published less than five months apart.
- MV-DNN model [15]: The core of the MV-DNN is a deep neural network (DNN) which is applied to several views. Its structure is essentially the same as that of a Deep Semantic Similarity Model (DSSM) [16], the difference is that the number of branches of the model can be adapted to the number of views of the data rather than being limited to two. Each view passes through a different DNN and the similarity between the outputs of each DNN is measured using a cosine similarity function. DSSMs are usually used on textual data but can be adapted to other types of data. Their architecture can be linear as in [15] or use another type, such as a convolutional structure in [16].

- TDSSM [17]: This model introduces a temporal dimension to the DSSM, it is called Temporal DSSM (TDSSM). The temporal dimension is added to the structure with a recurrent neural network (RNN), which makes the model capable of handling temporal views in addition to static views. In its application on a recommender system the user and product characteristics are passed as different views and the outputs of the neural networks are combined and compared with a cosine similarity function. This underlines the fact that the structure of a DSSM is fully modular and can be modified at will to meet data requirements.
- MV-AFM model [18]: The Multi-View Attentional Factorization Machines (MV-AFM) model is a neural network that uses hierarchical attention mechanisms at the feature and view level. It is composed of an embedding layer that allows the generation of dense and smaller vector representations of the data. The feature-level attention mechanism allows the embedding vectors to be weighted for each view. The next layer models pairwise interactions between the views by calculating the sum of Hadamard products between the vectors of each view. This extends the number  $n$  of views by  $\frac{n(n-1)}{2}$  interaction views. The attention mechanism at the view level then weights the new generated views to pay more or less attention to each view according to their relevance.
- NAML model [19]: This model is called News recommendation Approach with attentive Multi-view Learning. It uses a news article encoder with multi-view learning. This encoder is used to create an abstract representation of news articles. Hierarchical attention mechanisms are used within the encoder for features and views. Another attention mechanism is also used to combine representations of the multiple articles the user has viewed.

In this paper we will present some of these state-of-the-art models to apply them to our industry data and compare their performance in the context of our recommender system.

#### D. Attention mechanisms

The first attention mechanism in deep learning was introduced in 2014 in [20]. It was initially developed to improve the performance of encoder-decoder models over long sentences.

Encoder-decoders are composed of two recurrent neural networks, the first one encodes a sequence into a latent vector representation, this vector is called a context vector. The second uses this abstract representation of the initial sequence to generate a new output sequence. In such a model, the terms of the input sequence are propagated one after the other through the network. After several terms have been propagated, the model tends to forget the oldest terms because of the problem of vanishing gradient. The input sequence is encoded within a context vector, this vector is an embedding of fixed size and the decoder only has access to it to generate the new sequence. If some of the information has been forgotten or is missing within the context vector, then the quality of the output sequence will not be optimal.

To solve this problem, Bahdanau et al. [20] proposed an encoder-decoder in which the decoder not only relies on a single context vector to create the new sequence, but rather generates a new context vector containing the information needed at each step. In this model the decoder has access to all the hidden vectors of the encoder ( $h_1, \dots, h_n$ ). To generate each term of the new sequence the decoder relies on the last generated term  $y_{i-1}$ , on the previous decoder hidden vector  $s_{i-1}$  and on a context vector  $c_i$  capturing the relevant information of the input sequence for step  $i$ . A formal notation of the generation of the  $i$ -th term  $s_i$  by the decoder is the following recursive function :

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (2)$$

The context vector  $c_i$  captures the relevant information for the  $i$ -th decoding step within the hidden encoder vectors and the previous hidden decoder vector. A score  $e_{i,j}$  is calculated for each hidden encoder vector  $h_j$ , this score will be used to weight each hidden vector according to its relevance for the  $i$ -step:

$$e_{i,j} = a(s_{i-1}, h_j) \quad (3)$$

In this notation  $a$  is an alignment model, such as neural network layer. The parameters of the alignment model are trained simultaneously during the optimization of the global model. The obtained sequence of scalars  $e_{i,1}, \dots, e_{i,n}$  is normalized with a softmax function whose terms are defined as follows:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^n \exp(e_{i,k})} \quad (4)$$

The obtained vector  $\alpha_i$  is called the alignment vector and is used to weight the hidden encoder vectors to generate the context vector  $c_i$ :

$$c_i = \sum_{j=1}^n \alpha_{i,j} \cdot h_j \quad (5)$$

This vector contains the relevant information from the input sequence for the  $i$ -th decoding step.

Attention mechanisms can be applied at different levels in order to extract interesting and relevant information at each of these levels [21]. This theory is called hierarchical attention and has been successfully applied to capture basic notions about the structure of a document. Documents have a hierarchical structure, with words forming sentences that form paragraphs that form a document. Not all words have the same importance within a sentence and therefore should not be considered equal by the model, the same applies to the importance of sentences and paragraphs. A model that includes a hierarchical attention mechanism is therefore able to take into account this notion of various importance at different levels.

In this paper we will first describe multi-view learning methods applied to recommender systems which will be evaluated on our industry data in a comparative study. We will then detail the used data as well as the experimental protocol used for the study. Finally, we will present the results obtained in our comparative study.

## II. MULTI-VIEW LEARNING FOR RECOMMENDATION

### A. Available views

Rezulteo is an online comparison tool that contains a recommender system, this system is subject to constraints, such as the cold-start problem [4], implicit interactions between the user and the products, and no purchase confirmation after product recommendation (the user is redirected towards retailer sites).

This section focuses on the detailed presentation of the five available views for the Rezulteo project and on their individual evaluation. All the models that will be presented will exploit all or a part of those data sources to produce their recommendation.

- 1) User sessions data: This data is obtained from the user's browser session history. It is sequential data, since it brings together all actions taken by one user after another, in chronological order, thus its temporal dimension is primordial.
- 2) Expert product data: This is business data collected from manufacturers that characterizes each Rezulteo product. It is static data that associates a unique and abstract representation to each product.
- 3) Latent user vectors: Latent user vectors are generated by factoring matrices on user interactions using the eALS model, Element-wise Alternating Least Square model [22]. The goal is to reuse one of the two components (user matrix) needed to reconstruct the implicit interaction matrix to generate a unique and abstract representation for each user.
- 4) Comparability view: This static view is obtained from similarity coefficients which are calculated between each pair of products. These are determined by an expert system based on business indicators (age of the product, market sales volume and its presence on the different market shares, etc.). To make it exploitable, a vector of size  $n$  is associated with each product,  $n$  being the number of products in total. Each product is assigned a unique index between 0 and  $n - 1$ , which is determined arbitrarily and represents the position of the product within the list. The vector associated with each product contains the similarity values between that product and all other products in the system. The values are organized within the vectors according to the indices of the corresponding products in order to make it possible to learn about this data.
- 5) Compatibility view: This static view lists all the products that are compatible with user sessions, each session being associated with a query that determines this compatibility. As each vehicle is compatible with only a limited set of tires, this view simply represents the list of products that are compatible with the user session. It is a simple vector of boolean values of size  $n$ , where  $n$  is the total number of products.

The quality of each view has been analyzed by training a neural network on each isolated view. The architecture of the

used neural network is always the same, so that if the obtained recommendation quality is higher than the one obtained using randomly generated data, it means that the isolated view contains useful information for recommendation.

By using several views, all of which provide useful information, a deep composite model is able to naturally make use of the complementary principle. This principle defines that multiple views that all contain useful and complementary information can be exploited to lead to better learning performance than if they were treated independently of each other. A deep composite model using multiple views is also capable of making use of the consensus principle. This principle defines that by minimizing the disagreement between each view, the error rate on each view will also be minimized. Such a model is capable of using this principle since each view is associated with an independent branch in the model architecture. Thus, each branch learns from one view, then all the branches are combined into one. In the final branch, the abstract representations of each view are brought together to retrieve all the useful and complementary knowledge they contain and exploit them together to maximize learning performance.

To evaluate our views we used two common metrics in the recommendation field which complement each other :

- 1) HR (Hit Rank), it measures the system capacity to provide all relevant solutions (varies from 0 to 100%).

$$HR = \frac{hits}{users} \quad (6)$$

$hits$  : the number of users for which the product has properly been recommended.

$users$  : the total number of users.

- 2) NDCG (Normalized Discounted Cumulative Gain), it measures the quality of the produced ranking (varies from 0 to 100%).

$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)}$$

$$IDCG_p = \sum_{i=1}^{rel_p} \frac{2^{rel_i}}{\log_2(i+1)} \quad (7)$$

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$

$p$  : The position in the ranking for which the gain is calculated.

$rel_i$  : The graded relevance of the result at position  $i$ .

Table I shows the quality of each available view for this project in comparison to a view composed of random data. Each value is obtained by the mean of 10 experiments of 20 epochs each. The metrics are both calculated on a top 100 and all values are converted to percentages.

The user sessions data cannot be processed with the same neural network architecture as the other views, as it is sequential data. The session view is therefore evaluated using a 3D CNN (3D Convolutional Neural Network), a convolution model that is able to handle temporal data using three-dimensional convolutions, it has already been used for this

TABLE I  
EVALUATION OF EACH AVAILABLE VIEW IN COMPARISON TO A RANDOM VIEW.

Evaluated view	HR@100	NDCG@100
Random	66.33 ± 0.08	19.83 ± 0.04
User sessions	82.01 ± 0.30	32.42 ± 0.36
Expert data	83.58 ± 0.10	34.16 ± 0.13
Comparability view	83.83 ± 0.13	31.82 ± 0.13
Compatibility view	85.37 ± 0.17	33.89 ± 0.18
Latent user vectors	89.56 ± 0.10	37.21 ± 0.18

purpose by the authors of [7]. The used confidence interval is the standard deviation.

The evaluation of the quality of the views shows that all available views provide useful information to make recommendation. As the evaluation is based on a top 100, the random view obtains a HR and NDCG that might seem high, but we can see significantly better results for the other views. Deep composite models will therefore be able to exploit the consensus principle on the different views to obtain the best possible results.

### B. Evaluated models

This section focuses on the different state-of-the-art models that we have implemented, adapted and applied to our industrial data within the framework of the comparative study. Our goal is to study multi-view learning performance when applied to the field of industrial data recommendation. We chose three state-of-the-art models that will be compared to our Baseline, a hybrid between eALS and a 3D CNN. The model architectures are described and presented in a generic way in order to allow their application on different data.

Some of the models previously presented could not be applied to our industrial data or were too similar to other implemented models, thus they were not selected to be part of our comparative study.

1) *Baseline*: The used Baseline is based on the work of [23] from which we used the "hybridization by feature combination" to build this model. Its architecture is composed of a 3D-CNN, a convolutional model capable of handling sequential data. The input to this model is a concatenation of the user sessions data, expert data and latent user vectors. Therefore, it only makes use of three out of the five available views. The views concatenation is carried out so as to keep the temporal dimension of the user sessions data.

2) *MV-DNN, Multi-View Deep Neural Network [15]*: This model is capable to handle data organized in multiple views, it is composed of as many parallel branches as there are used different views. Each branch of the model handles one view and builds a dense and abstract representation of that view. The model possesses a pivot view (corresponding to the users data in the paper [15]) and aims to maximize the sum of the similarities between the pivot view and all other views by using a cosine similarity function.

For the sake of generalization and adaptation to our project, the implemented version of this model, showed on figure 1,

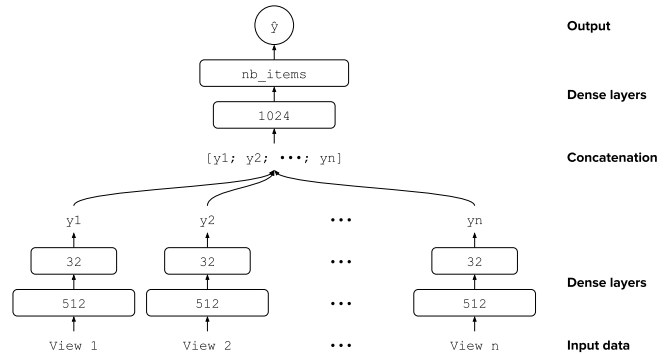


Fig. 1. Architecture of our implemented version of the MV-DNN.

does not make use of a pivot view and a similarity function. In our version the results obtained from each branch are concatenated to be used as input to a final neural network. The output of the model is a vector associating a recommendation weight to each ReZulteio product.

The developed model architecture is generic and can handle as many views as necessary. All the branches are made up of forward propagation neural networks with the same architecture, so each branch is able to manage one static view. The user sessions view is composed of sequential data, therefore it cannot be exploited by this model.

3) *TDSSM, Temporal Deep Semantic Similarity Model [17]*: The previously presented MV-DNN has the inconvenience of only being able to manage static data. In order to make use of the total available data for this project it is mandatory for us to use a model capable of handling both static and sequential views.

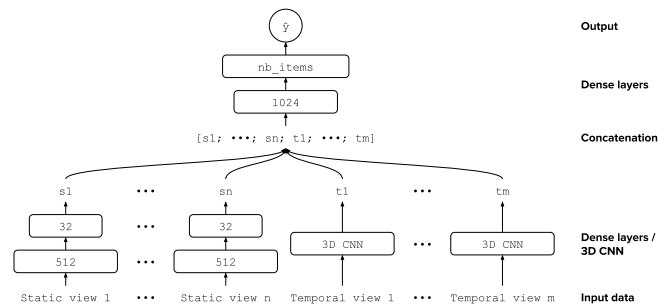


Fig. 2. Architecture of our implemented version of the TDSSM.

The TDSSM model extends the MV-DNN architecture by integrating one or more recurrent models within its branches. Rather than using a recurrent model, our implementation uses a 3D-CNN. As for the previous model, the implemented architecture is generic so that it can manage static and sequential views, it is represented on figure 2.

4) *MV-AFM, Multi-View Attentional Factorization Machines [18]*: The MV-AFM model introduces two new concepts to multi-view architectures, the hierarchical attention concept and the concept of pairwise interactions between the

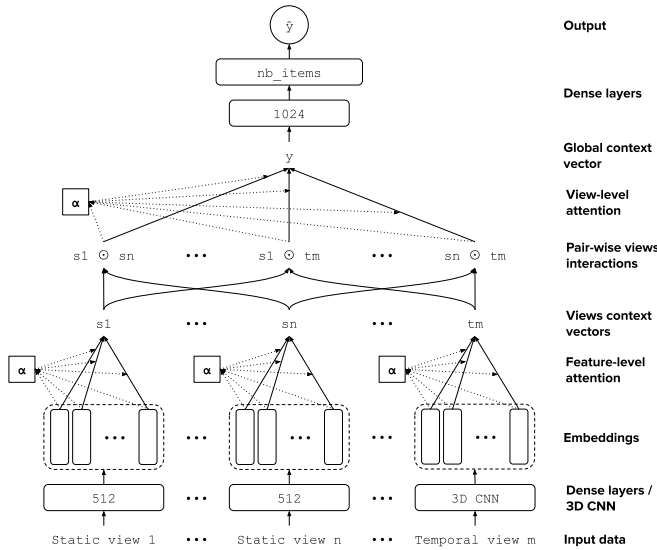


Fig. 3. Architecture of our implemented version of the MV-AFM.

views. The purpose of those two additions is to improve the learning capabilities of the model. Using attention mechanisms is an increment to the work previously done on the Rezulteo project in order to better manage the multi-view context. The pairwise combination of the views can theoretically allow correlating information between the views and thus improve recommendation performance. In the initial paper [18], the model was applied in a multi-view context on datasets from Google Play and from the Apple App Store. The model architecture has been slightly modified to be fully adapted to the Rezulteo project, it is shown on figure 3 and described more precisely below.

The model architecture is organized hierarchically, with a feature-level attention mechanism and a global view-level attention mechanism. There is one branch per view, each branch starts with a model specified to handle the input data type (static or sequential), which returns a fixed size vector. That vector goes through an embedding layer, which generates  $x$  vectors of size  $y$  by passing the input vector through  $x$  linear layers in parallel, each of  $y$  neurons. Those  $x$  vectors go through the feature-level attention mechanism which generates a context vector that theoretically contains the most relevant and useful information from each of the  $x$  entry vectors. The next layer models pairwise view interactions, it operates a Hadamard product between each context vector pair, which makes it possible to model the interactions between all pairs of views one by one. The view-level attention mechanism can then generate a global context vector that brings together all the relevant information from all the views and all their interactions. All branches are thus gathered into one and the final output of the model is obtained through two final linear layers.

The feature-level attention mechanism takes the matrix  $F$  as input,  $F$  is composed of the  $x$  vectors of the size  $y$

previously generated by the embedding layer. The purpose of this attention mechanism is to extract the most relevant knowledge from the input in order to generate a context vector from it. The realized experiments on this model have shown better performances with this attention mechanism than without. The mechanism can be formalized by the following 8.

$$\begin{aligned}
 e_i &= q \cdot \tanh(W \cdot F_i + b) \\
 \alpha &= \frac{\exp(e_i)}{\sum_{k=1}^x \exp(e_k)} \\
 c &= \sum_{i=1}^x \alpha_i \cdot F_i
 \end{aligned} \tag{8}$$

With  $alpha$  the alignment vector of size  $x$  and  $c$  the generated context vector of size  $y$ . The variables  $q$ ,  $W$  and  $b$  are weights and biases that are learned during the global training of the model.

The pairwise view interaction layer takes as input the context vectors obtained with the feature-level attention mechanism from each branch, thus there is one context vector per input view. The point of this layer is to model the interactions between each pair of views. The resulting number of views is  $o + \frac{o \cdot (o-1)}{2}$ , with  $o = n + m$ ,  $n$  being the number of static input views and  $m$  the number of temporal input views. Each interaction view is generated by the sum of the Hadamard product between two context vectors. The size of the output vectors is the same as the size of the input vectors,  $y$ . The output of this layer are the  $o$  input vectors and the  $\frac{o \cdot (o-1)}{2}$  newly generated vectors, on figure 3 the  $o$  input vectors do not appear in the layer output for the sake of clarity.

The view-level attention mechanism extracts the most useful and pertinent knowledge from all the views and their pair interactions. It takes as input the  $o + \frac{o \cdot (o-1)}{2}$  vectors from the previous layer and outputs one context vector of size  $y$ . This attention mechanism is based on the same model as the feature-level one and can therefore be formalized in the same way 8. The generated context vector is then given as input to two last forward propagation layers that output the final recommendation vector.

### III. INDUSTRIAL DATA AND EXPERIMENTAL PROTOCOL

We conducted our experiments on real data from an online tire comparison tool. We have two datasets at our disposal. The first one, noted D1, is composed of four views (user sessions data, latent user vectors, expert data and the comparability view), only the compatibility view is missing. The second one, noted D2, is a subset of the first one and is composed of the five available views. The compatibility view, missing from D1, is built for D2 by re-executing user queries to obtain the list of all the products that are compatible with the query. Table II summarizes the volume of the two datasets:

From the datasets volume it can be seen that there is about one session per user, which means that users usually only use the recommender system once, and therefore it is impossible for us to create a user profile. Without user profiles our

TABLE II  
DATASETS VOLUME.

Dataset	Sessions	Products	Interactions	Users
D1	114, 359	7, 726	307, 231	102, 613
D2	50, 241	3, 268	144, 095	46, 148

recommender system is subject to the cold-start issue, which means that the system has to recommend product to a user for whom it has no previous information. Furthermore, the data that is collected during the system utilization is implicit interactions between the user and the system, thus in our case it is not possible to rely on explicit user ratings or feedback as is usually the case with recommender systems. It should also be noted that the system does not collect information regarding the purchase of a product by the user, since the user is redirected to the merchant site if he is interested in a product, which further complicates the recommendation.

Our goal is to evaluate the four selected model performances on our two available datasets. Our Baseline will serve as a benchmark for comparison. The metrics used to evaluate the models are the same as those used in the previous evaluation of the views: HR and NDCG. Each interaction between the user and Rezulteo concerns a product, the target variable that the models learn to recommend is the product concerned by the next user interaction.

In order to keep a consistent comparison between the two datasets, we balanced the size of the ranking to be recommended depending on the number of products included in each dataset. The dataset D1 contains about 2.36 times as many products as the second, thus models using D1 will be evaluated on a top 236 and a top 12, while those using D2 will be evaluated on a top 100 and a top 5.

#### IV. COMPARATIVE STUDY

Table III shows the number of trainable parameters for each selected model, depending on the used dataset, therefore depending on the number of products that can be recommended.

TABLE III  
COUNT OF TRAINABLE PARAMETERS PER MODEL AND PER DATASET.

Dataset	Baseline	MV-DNN	TDSSM	MV-AFM
D1	8, 094, 087	12, 036, 431	12, 187, 687	10, 490, 119
D2	3, 524, 637	6, 907, 973	7, 059, 229	5, 752, 477

Tables IV and V show the performance of the models evaluated on datasets D1 and D2. Bold values are the highest values for each metric. Each value is obtained by the mean of 20 experiments for which the seed of the stochastic gradient descent varies. Each experiment is performed for 50 epochs for the Baseline and 10 epochs for the other models. Indeed, multi-view models converge faster than the Baseline, thus the training time of those models is shorter.

From these results it can be observed that the three models using the multi-view approach obtain better results than the Baseline. This shows that the multi-view approach is relevant

TABLE IV  
MODELS EVALUATION USING DATASET D1.

Metric	Baseline	MV-DNN	TDSSM	MV-AFM
HR@236	92.12 ± 0.13	93.45 ± 0.14	93.39 ± 0.16	<b>93.47 ± 0.21</b>
HR@12	49.04 ± 0.40	51.36 ± 1.50	<b>52.02 ± 1.02</b>	51.11 ± 0.65
NDCG@236	34.64 ± 0.21	<b>35.12 ± 0.97</b>	35.06 ± 0.76	34.55 ± 0.36
NDCG@12	26.36 ± 0.26	26.97 ± 1.23	<b>26.98 ± 0.96</b>	26.29 ± 0.43

TABLE V  
MODELS EVALUATION USING DATASET D2.

Metric	Baseline	MV-DNN	TDSSM	MV-AFM
HR@100	88.51 ± 0.15	91.79 ± 0.17	91.82 ± 0.12	<b>91.91 ± 0.22</b>
HR@5	33.18 ± 0.28	40.13 ± 0.26	40.22 ± 0.38	<b>40.33 ± 0.54</b>
NDCG@100	35.30 ± 0.17	39.84 ± 0.16	<b>39.87 ± 0.17</b>	39.57 ± 0.38
NDCG@5	21.93 ± 0.22	27.09 ± 0.22	<b>27.13 ± 0.31</b>	26.78 ± 0.49

and effective for our project. We can notice that the MV-AFM model obtains slightly better results than the others for the Hit Rate metric, and that the TDSSM gets better results for the NDCG metric. It may therefore be appropriate to use the MV-AFM if one wishes to maximize the chances of recommending the most coherent product and to use the TDSSM if one wishes to achieve a more relevant ranking. However, given the very narrow range of performance between the three multi-view models it is not possible to say for sure that one of those models is superior to the others.

In order to be able to distinguish the recommendation performance of the three multi-view models and to investigate why the MV-AFM seems to perform better on the Hit Rate and the TDSSM on the NDCG metric, it would be necessary to conduct a more thorough study that would compare their performance on various datasets and not only on a real case as was the case in this study.

Figures 4 and 5 show the attention level associated with each view and views interaction within the MV-AFM model, averaged over 1000 predictions. The y-axis represents the attention level in percentage and the x-axis shows the views and their interactions. The views are noted as follows:

- 3d : User sessions view, handled by the 3D-CNN.
- uv : Latent user vectors.
- cd : Expert data.
- cr : Comparability view.
- ct : Compatibility view.
- Interactions between views are noted “view 1 x view 2”.

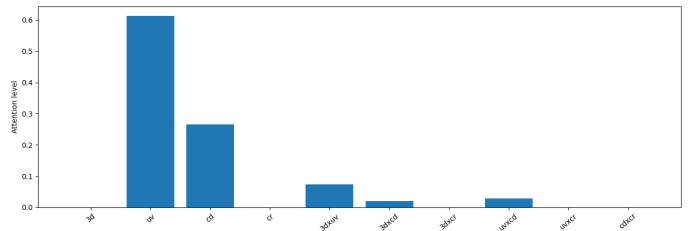


Fig. 4. Average attention levels for the MV-AFM on dataset D1.



