Final paper

ProVe - Self-supervised pipeline for automated product replacement and cold-starting based on neural language models

Andrei Ionut Damian
XCS224U Spring 2020

Abstract

In past years we have seen a lot of cross-domain application of various deep learning areas to real-life use cases, and in some cases, there was little to no apparent connection between the two (say neural language approaches applied to stock replenishment prediction). One of those areas is that of applying deep representation learning based on neural language processing to business analytics system. In our paper we will propose an efficient pipeline approach to several problems in the area of product recommendation systems, based on Natural Language Understanding. This document will try to revise several approaches in the area of Natural Language Understanding as well as work relevant to the current state of the art in the area of recommender systems based on neural language processing and representation learning in general. Based on this cross-domain approach we propose the hypothesis of inferring product replacement retail and distribution systems as well as address the problem of cold-starting product embeddings for products that newly arrive in existing inventories. The whole research and experimentation process has been done using real life private transactional databases, however the source code is fully available at: https://github.com/andreiionutdamian/ProVe

1 Introduction and problem definition

Deep representation learning is arguably one of the most important areas of machine learning as it allows us to automatically – and potentially in un-supervised fashion – discover rich features of the raw data. Since the introduction of the well-known word-embeddings generation algorithms in early 2015, the area on natural language processing and understanding has seen a tremendous improvement. Although it might seem that the actual state-of-the-art resides strongly on contextual embeddings and/or complex multi-head self-attention architectures, it is all based on the initial "basic" steps in the area of semantic vector space models.

Based on this rich area of research and experimentation a lot of development has been done in the area of applying NLP/NLU approaches to commercial transaction systems including recommender systems. One the most common approaches, further described later in section 2, is to view a retail transaction (a list of purchased items) as a natural language object – a context window, the whole retail transactional database as the actual text corpus and the item SKU database as the "word vocabulary". This approach of creating a hypothesis similarity between the natural language representation learning and business analytics systems has proved successful in various cases and it is still a very active area of research.

The main intuition behind our initial proposed deep learning pipeline was that we could, at least in theory, generate, through multiple modeling iterations, powerful-enough semantic vector space

embeddings for each individual item (product) so that we can infer replacements (synonym) items and propose them in the case on original item shortages – all of these in a self-supervised setting

Another important aspect we aimed to achieve in our work was to propose an efficient and scalable approach that would fully take advantage of GPU computational resources and that would easily scale for rea-life projects with large transactional databases and inventories of hundreds of thousands of items.

2 Related work

In order to summarize our proposed related work, it is best that we directly review the following list of objectives that fueled our project inception phase:

- ✓ Review semantic vector space model generation based on the most commonly known methods followed by understanding what tools and approaches we have at our disposal in order to fine-tune de embedding spaces.
- ✓ Review cross-domain application of NLU in the product recommendation systems that are relying on representation learning as well as understand

In the following sub-sections, a deeper look will be given at various approaches and "tools" we plan to employ in our proposed experiment. We will analyze a mix of fundaments of deep representation learning including but not limited to semantic vector space model generation as well as representation learning based recommender systems. As our target is to clearly define our related concepts and approaches, we will present each approach in a general perspective and also pinpoint our take-aways.

2.1 Semantic vector space models

One of the most important papers relevant to our subject is *"GloVe: Global Vectors for Word Representation"* (Pennington, Socher, & Manning, 2014) as it describes one the core representation learning methods required to generate sound semantic word embeddings beside the similarly well-known *word2vect* (Mikolov, Chen, Corrado, & Dean, 2013). The main intuition behind *GloVe* is based on the fact that statistics of word co-occurrence in a corpus is the primary source of information of the proposed unsupervised method for learning word representations. Thus, the proposed word vector generation pipeline first constructs the matrix of co-occurrence then applies a weighted least squares regression using as target the natural logarithm of the word counts as defined by the below objective function.

$$J = \sum_{i,j=1}^{V} f(X_{ij})\ (w_i^T \tilde{w}_j\ +\ b_i + \tilde{b}_j - logX_{ij})^2 \tag{1}$$

Finally, as the model generates through the optimization process to sets of word vectors (W and $\tilde{W}$), the *GloVe* algorithm summarizes the two matrices in order to obtain the final semantic vector space models.

Several other aspects of interest can be found in the paper for the construction of our hypothesis and

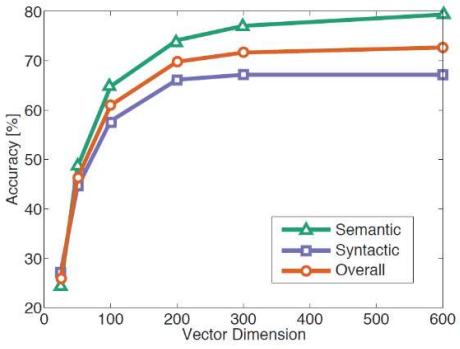


Figure 1 - Accuracy on a specific task as a function of vector size. Image from original paper by Pennington et al

one of the most important is the correlation between word vector dimension (and thus model capacity) and the task-related accuracy (see Figure 1).

2.2 Product recommendations

The logical next step in our related work review would be to analyze one of the early research papers that proposed the encoding of item SKUs in a similar manner to that of words

Grbovic et al proposes in the paper "*E-commerce in your inbox: Product recommendations at scale*" (Grbovic, et al., 2015) a direct approach of constructing product

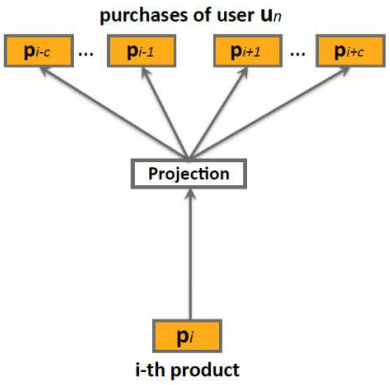


Figure 2 - The skip-gram architecture similar with original word2vec paper. Image from original paper by *Grbovic et al*

embeddings based on word2vec skip-gram model.

The authors propose a direct analogy between a certain product (item) in a basket and the focal word in the skip-gram context as presented in Figure 2.

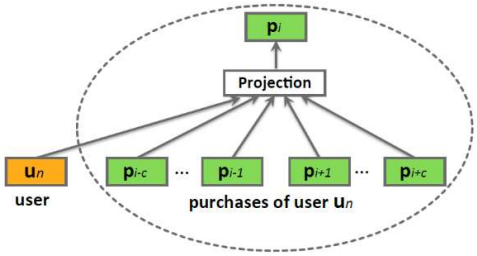


Figure 4 - Bag-of-words like approach to enrich the vector space model with user meta information. Picture based on the original paper by Grbovic et al

Another proposed aspect was that of enriching the vector space model with meta information such as user-identification in a similar method with *doc2vec* approach – a follow-up of the original word2vec – by Le and Mikolov (Le & Mikolov, 2014). In Figure 4 is presented this particular approach, this time being based on the BOW algorithm (predicting a focal item based on the context) instead of the skip-gram (predicting the context starting from a focal item).

In terms of actual vector space model applications, the authors propose two different hypotheses:

- straight correlation between items cosine distances and the actual real-life similarities of the products (namely the **prod2vec-topK** approach)

- a second more elaborate heuristic approach of finding viable "complementarity" baskets of products that might be sold well together. This second proposed predictive model, called **prod2vec-cluster** by the authors, leverages the hypothesis that similar items can be grouped together using distance metrics with K-Means algorithm. Furthermore, close clusters generated in this manner yield potentially well correlated products and picking items from different close clusters will generate a well-defined complementarity basket.

This particular work is one of the earliest and most well known in the area of applying representation learning and neural language modelling to the problem of retail recommender system.

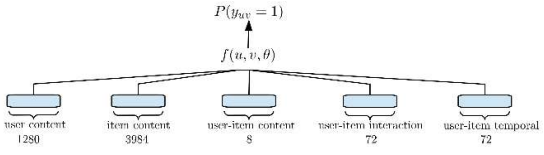


Figure 3 - Figure from paper of Volkovs et all presenting the overall architecture of their classification model using over 4000 hand-crafter features

In contrast with this work, the ACM Recommender Systems Challenge'17 was won by a more recent yet more conservative approach by Volkovs et al. In their paper "*Content-based Neighbor Models for Cold Start in Recommender Systems*" (Volkovs, Yu, & Poutanen, 2017) the authors propose a classic approach based on heavy manual featurization of the input data and full supervised approach. Although the proposed method received praises due to the high scores in the competition it is obvious that this method relied entirely on heavily supervised dataset and hand-crafted features and does not possess the flexibility and the scale ability potential of self-supervised system.

2.3 Retrofitting and counter-fitting

Following the analysis of the base tool – the *GloVe* word-embeddings generation approach and its particular (and potential) cross-domain application to recommender systems - it is now required that we revisit research work in the area of enriching vector space models based on retrofitting and counter-fitting. For this subject we decided to analyze three important papers: "*Retrofitting Word Vectors to Semantic Lexicons*" by Faruqui et al (Faruqui, et al., 2014), "*Counter-fitting word vectors to linguistic constraints*" (Mrkšić, et al., 2016) and the more recent work of Benjamin J. Lengerich, Andrew L. Maas and Christopher Potts "*Retrofitting Distributional Embeddings to Knowledge Graphs with Functional Relations*" (Lengerich, Maas, & Potts, 2017).

Basic retrofitting

In order to better understand the motivation for the review of vector space models retrofitting approaches (and in a later section the review of the counter-fitting approaches) we have to revisit the work done by *Grbovic at al* as well as other teams that pursued the goal of obtaining products semantic vector space models. In all above approaches the authors base their work on the shallow hypothesis that similar items (*words*) measured by cosine distance might have a synonym-like or an entailing-like relationship. However, this as possible for product vector space models as it is with natural language vector space models counterparts. For example, a *GloVe-300* vector space model, pretrained on *Wikipedia* and *Gigaword* might tell us that *'adolescent'*, *'teenager'*, *'puberty'* are very similar to each other based on their pairwise cosine distance and actually '*adolescent'* is more similar to '*puberty'* than it is to '*teenager'*. Exactly the same stands with product semantic vector space models – just using a raw embedding generated with *GloVe* or *word2vect* is not enough to get rich and reliable properties of the items.

This is the point where adding meta-information similar to the *synonymity graphs* in natural language might greatly aid us in further refining the semantic power of our product vector space model. The main intuition is well defined in the work of Faruqui et all as "*graph-based learning*

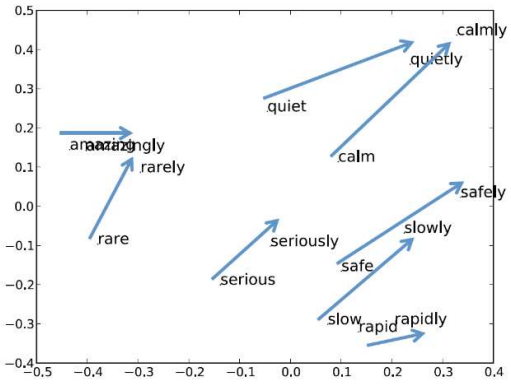


Figure 5 - Closing the distance between adjective-adverb pairs. Figure from original paper by Faruqui et al presenting the post-retrofit vector space projection

technique for using lexical relational resources to obtain higher quality semantic vectors" by means of post-processing the pre-trained semantic vector space models.

$$J = \sum_{i=1}^{N} \left[\alpha_i \|q_i - \hat{q}_i\|^2 + \sum_{i,j \in E} \beta_{ij} \|q_i - q_j\|^2 \right] \quad (2)$$

Another important aspect is that the proposed "*retrofitting*" optimization method, described by the objective function in equation 2, is agnostic to the origins of the input vector space model, such as

original training objective or overall initial training approach. In the this equation Q is the new optimized vector space matrix, $\hat{Q}$ is the original vector space model, and *i,j* are pairs of words that are connected by an edge in *E*.

As the authors explain in the paper, they perform experiments using various semantic lexicons such as PPDB (Ganitkevitch, Van Durme, & Callison-Burch, 2013) and WordNet (Miller, 1995), use these in order to improve the word vectors. Following the optimization process they evaluate the quality of the new *retrofitted* word vectors in order to determine how well they capture semantic aspects.

In another paper on the subject of counter-fitting (Mrkšić, et al., 2016) the authors take a more generalized approach by deciding to intuitively pull together the synonymous words while pushing antonymous words vectors apart. The pushing operation proposed by Mrkšić et *al* can be viewed as a similar operation to that proposed by the work of Faruqui et al although the optimization process is slightly different. *The counter-fitting* method proposes a three-part objective function as follows:

- the first component, described below in equation (3), of the objective function is responsible of antonym pushing by imposing a minimal *distance* δ (using a distance function *d*) for the antonym word embeddings (τ is *max* function)

$$J_A = \sum_{u,w \in A} \tau\,(0, \delta - d(v'_u, v'_w)) \quad (3)$$

- the second term – equation (4) - purpose is to pull synonym vectors closer by reducing the distance between them to a certain margin and it is closely related with the second term from the objective function in the Faruqui et al work described by equation (2).

$$J_S = \sum_{u,w \in A} \tau\,(0, d(v'_u, v'_w)) \quad (4)$$

- the third and final component presented in equation (5) has the purpose of preserving the overall vector space structure by minimizing the difference between the original vector space word embedding pairs and the new *counter-fitted* word embeddings. This particular term is quite similar – and has the same purpose - with the first term in the objective function proposed by Faruqui et all (Faruqui, et al., 2014)

$$J_{VSP} = \sum_{i=1}^{n} \sum_{j \in N(i)} \tau\,(d(v'_u, v'_w) - d(v'_u, v'_w), 0) \quad (5)$$

Finally, the full objective function in equation of the *counter-fitting* approach is simply the weighted sum of the three components as follows:

$$J = w_1 * J_a + w_2 * J_s + w_3 * J_{VSP} \quad (6)$$

Advanced retrofitting

Recent work of Benjamin J. Lengerich, Andrew L. Maas and Christopher Potts show that some of the core assumptions of the original base *retrofitting* approach of Faruqui et al, such as that connected entities should have similar embeddings, cannot hold for any real-life applications - such as health knowledge graphs. In order to address the underlined limitations, the authors propose *Functional Retrofitting* - a retrofitting approach that sees pairwise entity relations as functions rather than simple embedding straight similarities. One of the first implication of this approach is that we can include both the *pull* and the *push* between embeddings based on real-life similarity or dissimilarity in a more robust and generalized way that the one presented by *Mrkšić et al*. In order to further understand the approach proposed by the authors of the *Functional Retrofitting* let us understand the objective function in equation (7) by dissecting each of its four components. In order to simplify the explanations, we use a slightly different notation than in the original paper (Lengerich, Maas, & Potts, 2017)

$$J(Q, F) = J_{VSMP} + J_C + J_D + \sum_{r \in R} \rho_\lambda(f_r) \quad (7)$$

$$J_{VSMP} = \sum_{i \in Q} \alpha_i \|q_i - \hat{q}_i\|^2 \quad (8)$$

$$J_C = \sum_{(i,j,r) \in \mathcal{E}} \beta_{i,j,r} f_r(q_i, q_j) \quad (9)$$

$$J_D = \sum_{(i,j,r) \in \mathcal{E}^-} \beta_{i,j,r} f_r(q_i, q_j) \quad (10)$$

The first component J_{VSMP} of the objective function is the semantic vector space model preservation constraint, identical to that in (Faruqui, et al., 2014) and similar to equation (5) of (Mrkšić, et al., 2016). The second J_C and third J_D terms represent the application of relational penalty function for both the positive-relationship $\mathcal{E}$ knowledge-graph as well as for the $\mathcal{E}^-$ *negative space*.

Probably the most important difference between the work of (Faruqui, et al., 2014), (Mrkšić, et al., 2016) and the currently analyzed work of (Lengerich, Maas, & Potts, 2017) if the fact that we can use function-based relationship modelling. The authors of *Functional Retrofitting* propose in their paper two different approaches of the relation modelling – a linear relationship presented in Figure 6 and a basic fully connected neural network with one hidden layer.

$$\Psi_{\mathcal{G}}(\mathcal{Q}; \mathcal{F}) = \sum_{i=1}^{n} \alpha_i \|\boldsymbol{q_i} - \hat{\boldsymbol{q}}_i\|^2 + \sum_{(i,j,r) \in \mathcal{E}} \beta_{i,j,r} \|\boldsymbol{A_r q_j} + \boldsymbol{b_r} - \boldsymbol{q_i}\|^2 - \sum_{(i,j,r) \in \mathcal{E}^-} \beta_{i,j,r} \|\boldsymbol{A_r q_j} + \boldsymbol{b_r} - \boldsymbol{q_i}\|^2 + \lambda \sum_{r \in \mathcal{R}} ||\boldsymbol{A_r}||^2$$

Figure 6 - Linear relationship proposed by the authors of *Functional Retrofitting*. Applying *β=0* for *negative space* $\mathcal{E}^-$ and using identity as the value for the linear equation coefficient *A* gives us the objective function proposed by (Faruqui, et al., 2014) – image taken from original paper

The authors make the source code available at https://github.com/roaminsight/roamresearch.

3 Overall architecture

3.1 Self-supervised learning for the win

As already mentioned, the winner of the ACM Recommender Systems Challenge'17 presented in section 2.2 is entirely based on supervised data and heavy manual featurization. We believe that availability of data as well as relying on well annotated data and supervised datasets will become more and more inefficient for large scale systems deployment with good generalization capacity. Worth to mention is that in the particular approach the model devised by (Volkovs, Yu, & Poutanen, 2017) is not able to go beyond the clear and well defined purpose it was designed for and generate other insights such as map unknown relationships or generate rich semantic knowledge out of unsupervised data.

Regarding *retrofitting* methods we already observed the objective similarity between the work of (Faruqui, et al., 2014) and the (Mrkšić, et al., 2016) where the latter improves upon Faruqui et al objectives with the addition of *antonyms push.* More interesting seems to be the parallel analysis of the previously mentioned two papers and the work of (Lengerich, Maas, & Potts, 2017). We can view equation (9) in the *Functional Retrofitting* framework as the second term in the objective function from (Faruqui, et al., 2014) and respectively the J_S – equation (4) – from (Mrkšić, et al., 2016). Although equation (10) does not have an intuitive counter-part in equation (2) from (Faruqui, et al., 2014) we can consider that the *anonymity* pushing term, equation (3) from (Mrkšić, et al., 2016), is similar.

Maybe one of the most important take-aways of this particular paper (Lengerich, Maas, & Potts, 2017) is exactly the cross-domain application – starting from semantic vector space models and knowledge graphs in general and finally using the *Functional Retrofitting* approach to identify potential uses for existing drugs in new diseases where those drugs were not yet employed.

3.2 Two step pipeline

The two main work-horses of our pipeline are basically the initial semantic vector space

generator followed by the embeddings fine-tuning model.

For the task of generating the product embeddings out of transactional information we employ either *GloVe* or *word2vec* approach to vector space generation. In our experiments we will tried and succedeed to demonstrate that applying *GloVe* approach will yields similar if not better results than (Grbovic, et al., 2015) in obtaining basic semantic vector space representations of products.

While for word2vec embeddings we used the powerful and well-known *gensim* (Řehůřek & Sojka, 2010) package, for the particular *GloVe* implementation we used a extremely fast in-GPU approach that loads all co-occurrence statistics directly in the GPU computational graph and thus does not uses VRAM-RAM bus communications for batch submission, a approach forked from *Mittens* (Dingwall & Potts, 2018). Nevertheless, we have to mention that a 13,000 products inventory projected into a 128d vector space takes almost all the available 8GB VRAM of a GTX 2070 while it trains at 5 epochs per second.

Following the basic semantic vector space creation, we employ a retrofitting method using the existing meta-information in the retail databases – information such as **product categories** or even detailed category management tree-structures. This approach leads to the decrease in the cosine distance between products that can actually replace each other in real life in a similar manner as presented in the related work retrofitting papers address word vectors.

4 First things first: The Data

The data we used in our initial experiment was based on the raw extraction of transactional and inventory information from a retail system that has both brick-and-mortar as well as online operations. The real-life transactional dataset contains more than 2M transactions of a retailer with various locations over a period of more than 3 years. Although the dataset contains more than 15,000 different products in order to speed-up experiment time using full in-GPU training we reduced the number of products to a maximum of 13,000 top sold products.

Due to the size of transactional databases that often have billions of transactions each with many individual items we generate the proposed matrix of co-occurrence (MCO) with efficient batch reading of the transactional data.

The proposed experimental real-life data comes within a few files generated by SQL commands and exports from an existing ERP system of our retailer. The most notable data files are the transactional database file and the metadata file. The metadata information – presented in ***Table 1*** - taken directly from a real-life production system (ERP) contains raw information minimally describing each product-SKU `IdemId` with product name (`ItemName`) and other information such as number of item sales in observed in the selected period, a unique sequential item identificatory tag (`IDE`) as well as the hierarchy information in two fields Ierarhie1 and `Ierarhie2` that will be further used as a knowledge graph similar to WordNet (Fellbaum, 2012). There is also a secondary de-obfuscation data-source that contains for each hierarchy identified the actual name of that category. This information can also be used as a source for self-supervision in the process of creating the knowledge graph for the fine-tuning of our semantic vector space model.

The real-life transactional dataset, as previously mentioned, contains over 6M observations for over 2M different transactions. Each observation in the transactional database contains a `BasketId` identifier of the transaction as well as individual transaction detail information such as `ItemId` (and its counterpart `IDE`), a `SiteId` field that identifies the location of that particular transaction, a `TimeStamp` time-identification that is basically the same for all items of the same transaction, a quantity field, a customer identifier field and a product availability indicator.

Table 1 - Raw inventory information containing basic hierarchy information

ItemId	IDE	ItemName	Ierarhie1	Ierarhie2
406083	2	FELICITARI A 7331335123458	11	107
258901	6	TURTA DULCE TIP INIMIOARE	20	269

527499	13	TIBI USERIU / 27 DE PASI	1	9
493855	15	IRINA BINDER / FLUTURI VOL. 3	1	0
651356	23	SACOSA BIO CARTURESTI	11	100
522568	24	IRINA BINDER / INSOMNII	1	9
599979	30	MICHELLE OBAMA / POVESTEA MEA	1	9
439986	34	PAULA HAWKINS / FATA DIN TREN	1	0

4.1 Self-supervised training and supervised testing

Although the training process is entirely self-supervised, one particular challenge that was faced was the preparation of test cases and actual evaluation metrics. For the development of the evaluation dataset a classic "supervised" approach has been adopted based on manual data exploration. After manual selection of a limited list of categories based on the available meta information a sample of products has been extracted and analyzed. For each of the target selected categories - `MUSIC, VYNIL, DVD, Board-games` - multiple products have been manually selected. Following the manual selection of product candidates for evaluation data, for each individual chosen item the top neighboring items have been prepared using cosine distance.

4.2 Metrics and overall evaluation

With regard to metrics and evaluation of this particular experiment it is important to decide how we view the model - as a classic classification or as a ranking problem. As a result, the possible metrics that can be applied to our experiment are varied and the options range from the classification metrics such as accuracy, precision, recall, F-score up to recommender systems domain-specific metrics. The initially proposed metrics were Recall@K and MRR@k – recall@k that measures the ability of our system to recommend viable items within the first K proposed candidates as well as the more importantly the MRR@k (Mean Reciprocal Rank) applied only for the first K candidates (if the target is ranked lower than the K then the score is 0).

$$MRR = \frac{1}{|Q|}\sum_{1}^{|Q|}\frac{1}{r_i}$$

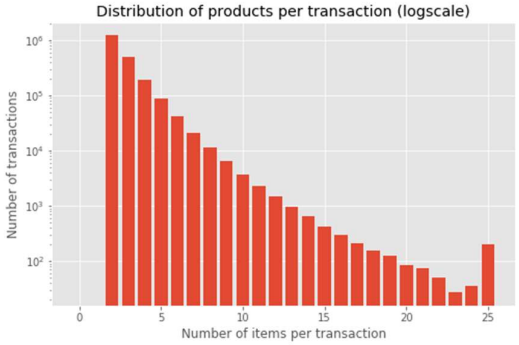
In the above formula Q represents all the tests performed while r_i represents the rank of the positive candidate (*inf* if no candidate matches the gold value). Finally, we decided to use only MRR as evaluation metric with various small number of candidates (K=1 and K=5).

5 NLU meets BPA - semantic vector space creation

In terms of natural language models analogy to Business Predictive Analytics systems, we consider each individual basket as a "sentence" and consider each individual product id as a "word". Referring to our proposed method of constructing matrix of co-occurrence counts and applying *GloVe* approach for the vector space representation modeling we have two options:

- ✓ using a specific context window size controlled as a model hyperparameter with or without weighting between focal and each individual item;
- ✓ considering each transaction as the overall context with no distance weighting between the focal word and the context word;

Regarding the context window it is important to note that for online shopping systems the items ordering, in each individual transaction, is quite important as it is directly correlated to the user journey in the website. As such for the particular case the definition of a hyperparameter that will control the context window size is important. For classic brick-and-mortar retailers the order of each product in the receipt is not important as it is unlikely that it reflects the actual user sequence of actions. In our particular case of the real-life dataset we have a retailer that has over 30 locations and a website that seems to generate less than 10% of the revenue. We decided to consider each transaction as a single-context for the computing of item co-counts.



With regard to the meta-data that can generate important knowledge graph information for our self-supervised approach we found out that the detail category - *or the so called "level 2 hierarchy"* - has a finer granularity than the first level hierarchical information. However, we decided to use the actual intersection of all categories as it is unknown if the hierarchy information is strictly tree-based in all retail systems.

As previously mentioned, the proposed *ProVe* model was expected to generate similar results with those of *GloVe* applied to word-vectors based on text corpuses. Just as we load a pre-trained GloVe-100 embeddings matrix with the 400,000 words vector space representations and run a neighbourhood analysis for a word such as "*beatles*" and obtain "*lennon*", "*mccartney*" we obtained similar outputs from our product semantic vector space model.

The hyper-parameter selection was entirely based on the results of our exploratory data analysis on the real-life dataset combined with the know-how resulted from the natural language counterparts. For the vector space dimensionality, we have chosen 128d. We chose a maximum frequency of co-occurrences of 250 although as it can be observed from the distribution of counts we have a maximum of over 5000 – nevertheless we found 250 as being the best setting during our experimentation process. The number of iterations was initially on-purpose set to a large number (250,000 epochs) followed by experimentation with an early-stopping mechanism based on cosine-similarity testing of product embeddings that led to much improved results.

Following the full generation of the ProVe vector space model for product embeddings we explored various neighbor tests in a similar manner to that of word embeddings. As previously mentioned we selected manually each individual evaluation item in order to find those that would clearly fail at proposing a good replacement candidate with just a cosine distance neighbor search within the products vector space.

Aside from the straightforward neighborhood search we also hand-picked two other products - one that is obviously a potential replacement for our product ("*Best of David Bowie - 2002*") and another one that is clearly not a replacement ("*MASINI DIE-CAST 7.5cm III ASST*"). For these two additional products we constantly measure the distances during the various phases of our experiment.

Table 2 - The top neighbors of David Bowie's "Blackstar" based on cosine distance

ID	DIST	NAME	H1
12071	0	CD / DAVID BOWIE / BLACKSTAR (	MUSIC
11418	0.643	CD / DAVID BOWIE / BEST OF BOW	MUSIC
4416	0.644	CD / COLDPLAY / 4 CD CATALOGUE	MUSIC
4865	0.663	CD / COLDPLAY / A HEAD FULL OF	MUSIC
9745	0.669	MASINI DIE-CAST 7.5cm III ASST	GAMES
7933	0.674	CUVINTE INCRUCISATE/ IQ FUN	BOOKS
8392	0.676	13.2X2.5CM PLATE	ACCESS
10135	0.678	BUCATARIE ITALIANA 100 DE RETE	BOOKS
4400	0.681	CARTEA SARUTURILOR / IV CEL NA	BOOKS
12266	0.697	TIM DEDOPULOS / HERCULE POIROT	BOOKS

In the previous example we see that a potential replacement for our proposed product `12071` is actually the closest product (`11418`) so this might seem a perfect fit, however we would like to have more similar products in the neighborhood of the target products and less items that clearly do not have anything in common with our product.

6 Fine tuning products semantic vector space

The second step in our pipeline is taking the initial product embeddings obtained using self-supervised optimization and further fine-tune them again with self-supervised methods using meta-data obtainable from the retail systems. For

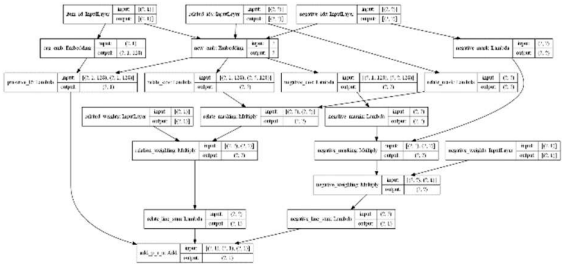


Figure 7 - Computational graph (same layout for *Tensorflow* and *PyTorch*) using cosine distance as the relation distance function

the fine-tune system we applied the several methods derived from the work presented in the related work section and finally arrived at a generic cost function presented below where $J_{preserve}$ represented the vector space preservation part of the loss functions denoted by the *distance* between the initial vector representation and the modified vector space representation, J_{relate} represents the objective function that pulls together the similar items based on the *membership* on the same meta-data categories denoted by the $\mathcal{E}_r$ graph. J_{negate} is represented by an objective function that uses negative relations within $\mathcal{E}_n$ in order to *push* apart non-similar product vectors.

$$J = J_{preserve} + J_{relate} + J_{negate} \quad (11)$$

$$J_{preserve} = w_p \sum_{i \in Q} D_p(q_i, \hat{q}_i) \quad (12)$$

$$J_{relate} = w_r \sum_{i,j \in \mathcal{E}_r} D_r(q_i, q_j) \quad (13)$$

$$J_{negate} = w_n \sum_{i,j \in \mathcal{E}_n} D_n(q_i, q_j) \quad (14)$$

During our experimentation we experimented with various distance functions D such as L1, L2 and cosine distance ($D_n = \max(m - dist(q_i, q_j), 0)$ for the case of the "push" objective function J_{negate}) for each of the three components of the objective function. The resulting computational graph is presented in Figure 7.

Following the optimization process we obtained results presented in Table 3 that are drastically improved versus the initial ones presented in Table 2.

Table 3 - Post retrofitting product vectors using hierarchy categories

DIST	NAME	H1
0.000	CD / DAVID BOWIE / BLACKSTAR (	MUSIC
0.121	CD / RED HOT CHILI PEPPERS / T	MUSIC
0.138	CD / LANA DEL REY / LUST FOR L	MUSIC
0.151	CD / QUEEN / GREATEST HITS (20	MUSIC
0.155	CD / COLDPLAY / A HEAD FULL OF	MUSIC
0.157	CD / QUEEN / THE PLATINUM COLL	MUSIC
0.163	CD / FLORENCE + THE MACHINE /	MUSIC
0.164	CD / QUEEN / BOHEMIAN RHAPSODY	MUSIC
0.165	CD / LED ZEPPELIN / REMASTERS	MUSIC
0.165	CD / PINK FLOYD / THE WALL (Re	MUSIC

7 Conclusions and further work

In order to apply our work to real-life experiments we decided to find the optimal tools that would greatly speed-up the particular step of retrofitting process. We used in our experiments both *Tensorflow* (Abadi, et al., 2016) as well as *PyTorch* (Paszke, et al., 2019) in order to find the optimal framework that would lead to both scalable models and fast execution. We found that *PyTorch* with its ability to execute the whole graphs in the GPU memory as well as iterate the datasets directly in GPU runs an optimization epoch in aprox. 3 seconds compared with 11

seconds *Tensorflow* in graph mode and over 20 seconds in *Tensorflow* eager execution mode. Both methods are available on the project GitHub address.

Further experimentation work is required at this point to tune and test the models as well as finalize production-grade version of the proposed `get_item_replacement` as well as the `cold_start_item` method. More research and experimentation are also required in the area of determining an optimal heuristic method for automatically choosing the re-weighing factors of the objective function components.

Authorship

This research and experimentation presented in this paper has been conducted by Andrei Ionut Damian – andrei@lummetry.ai and the source code is fully available on GitHub.

5 References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. *12th {USENIX} Symposium on Operating Systems Design and Implementation*, (pp. 265-283).

Dingwall, N., & Potts, C. (2018). *Mittens: An Extension of GloVe for Learning Domain-Specialized.* arXiv preprint arXiv:1803.09901.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., & Smith, N. A. (2014). *Retrofitting word vectors to semantic lexicons.* arXiv preprint arXiv:1411.4166.

Fellbaum, C. (2012). WordNet. *The encyclopedia of applied linguistics*.

Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2013). PPDB: The paraphrase database. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 758-764).

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., & Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1809-1818.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. *International conference on machine learning*, (pp. 1188-1196).

Lengerich, B. J., Maas, A. L., & Potts, C. (2017). *Retrofitting distributional embeddings to knowledge graphs with functional relations.* arXiv preprint arXiv:1708.00112.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781 .

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 39-41.

Mrkšić, N., Séaghdha, D. O., Thomson, B., Gašić, M., Rojas-Barahona, L., Su, P. H., & Young, S. (2016). *Counter-fitting word vectors to linguistic constraints.* arXiv preprint arXiv:1603.00892.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., & Desmaison, A. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, 8024-8035.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors forWord Representation. *Proceedings of the 2014*

conference on empirical methods in natural language processing (EMNLP), (pp. 1532-1543).

Řehůřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (pp. 45-50). Valletta, Malta.

Volkovs, M., Yu, G. W., & Poutanen, T. (2017). Content-based Neighbor Models for Cold Start. *In Proceedings of the Recommender Systems Challenge 2017*, (pp. 1-6).