Recurrent Neural Networks for Modeling Company-Product Time Series

Katsiaryna Mirylenka, Christoph Miksovic, and Paolo Scotton

IBM Research – Zurich,
Säumerstrasse 4, 8803 Rüschlikon, Switzerland
{kmi,cmi,psc}@zurich.ibm.com

Abstract. Given the increasing amount of data related to information technology (IT) inventories and products purchased by companies, it is advantageous to analyze this information and model the IT install base of companies. Such an analysis and modeling reveal latent connections between deployed IT products and helps to discover discriminative features of the IT install base structure. This allows one to efficiently compare products and companies, apply appropriate marketing strategies towards similar sets and recommend future products.
In this paper we verify the temporal correlation of product time series. We formalize the notion of a company in terms of its IT install base and study the applicability of language modeling techniques emerging in natural language processing to the task of company-product modeling. The analysis is done using n-gram models and Recurrent Neural Networks (RNN) over a corpus of more than 800k companies. We assess various configurations of the modeling techniques and select the best model using the perplexity measure.
The results of this study demonstrate that RNN with one hidden layer and product embeddings of size 200 is optimal for the company IT install base representation in terms of goodness of fit of the model. Additionally, RNN provides meaningful features for products, which can be used to focus marketing campaigns on specific install base structures and enhance a sales recommendation system.

1 Introduction

During the past decade, the amount of marketing intelligence data, provided by specialized companies, has been growing dramatically. This data has proven to be extremely useful to get market insights in several contexts such as, for example, new markets development or white space determination. For a given set of companies, the data provides insights into the type of IT equipment they own and how this equipment is distributed across their physical locations. Usually the data also contains timestamps related to the equipment acquisition and the confidence of its presence.

In this work, we first demonstrate the sequential property of our data by means of hypothesis testing. Then, we evaluate different architectures of Recurrent Neural Networks (RNN) for the task of company-product modeling. We choose the best model in terms of goodness of fit or best predictive power. This model is used to extract product representations that can be applied for similarity search queries and marketing recommendations.

The main contributions of this work are the following:

1. We formalize the problem of modeling company install bases considering time series of product appearances, such that state-of-the-art unsupervised techniques from Natural Language Processing (NLP) can be applied.
2. We experimentally verify the sequential nature of the product appearance time series by applying statistical hypothesis testing based on the binomial distribution.
3. We assess the applicability of language modeling via RNN to our data. We demonstrate that RNN with one hidden layer and product embeddings of size 200 fits the data best and, additionally, provides discriminative embeddings of products.

The paper is organized as follows. In Section 2, we show the details of the available competitive install base data and provide a formalization of our problem. Section 3 discusses related approaches and their applicability. The application of RNN is presented in Section 4. Finally, Section 5 provides the comparison between different RNN architectures and discusses learned product embeddings.

2 Preliminaries and Problem Formalization

As a base for this work, we rely on the install base information provided by HG Data Company, Inc. [4].

For each company assessed, the following information is provided: the type of IT products available at each site of the company[1], the confidence of the information, and first confirmation date of the product presence. Product descriptions are organized in a hierarchical fashion containing three levels. The highest level of the hierarchy contains *Category Parents*, giving a high-level description of the product type, for example, “Data Center Solution” or “Hardware (Basic)”, . The *Categories* level contains finer-grained descriptions, such as “Printers” or “Midrange Computers”. Finally, each *Category* contains the *Products*, which constitutes the lowest level.

In this study we chose to focus on the *Category* layer. Therefore for each company we consider the product categories[2] associated with a company. In our version of the HG Data Company database, there are 91 distinct categories. Out of those categories, we restrict our study to hardware and low-level hardware-management software categories. We end up with 38 distinct categories.

We create our corpus for model training based on a set of product category appearance time series, which will be referred to as product time series in the remainder of this paper.

[1] More precisely, the database reveals whether a given product type is present, but without providing quantities.

[2] In the remainder of the paper, we use the terms products and product categories interchangeably, meaning product categories.

More formally, we consider a set of N companies represented in the HG Data Company database $C = \{c_0, \ldots, c_{N-1}\}$. Each company c_i has a time series of products A_i of length k_i. These products belong to the set of all possible products $A = \{a_0, \ldots, a_{M-1}\}$. That is:

$$\forall c_i \in C \; ; \;\; c_i \longmapsto A_i = (a_{i_0}, \ldots, a_{i_{k_i-1}}) \subset A.$$

The products from A_i are sorted by the time of their appearance in the IT install base of a company.

Given this formalization, the goal of this paper is to model company-product time series to provide recommendations about the set of possible future products. Additionally, we would like to learn the most representative features of products $\mathscr{B}$, based on their temporal proximity. The features should be representative in terms of goodness of fit of the generative model of company-product data.

Sequential nature of the data. To assess correlations between the consecutive values of product time series, we use statistical hypothesis testing. We verify whether the frequencies of product n-grams are statistically significantly higher than in the case of independent (1-gram) product observations. Therefore, the significance test depends on the actual frequencies of the individual products, the number of observed n-grams, and the frequencies of n-grams. The test statistic shows whether the frequency of an n-gram could be generated randomly from 1-grams.

In this random 1-gram model, every symbol in the sequence is drawn independently from a categorical distribution. The probabilities of the individual symbols can be estimated from their occurrences in the company-product time series. We calculate the frequencies of each product $Fr(a_1)$, $Fr(a_2)$, ..., $Fr(a_M)$ and the corresponding probabilities $Pr[a_i] \approx Fr(a_i)/T$, where T is the total number of products over all company product time series.

The random generation is connected to the fact that all products of a sequence are temporally independent and distributed as Bernoulli trials with the corresponding probabilities observed in the time series. The probability of seeing a sequence of products $X = (x_1, x_2, ..., x_k)$ in this case is equal to $Pr[X] = Pr[x_1, x_2, ..., x_k] = Pr[x_1] \cdot Pr[x_2] \cdot ... \cdot Pr[x_k]$. The distribution of a frequency of a sequence X is binomial with the parameters T and $Pr(X)$: $Fr_X \sim B(T, Pr(X))$. In consequence, the frequency Fr_X of X is a random variable with known distribution that is based on the properties of the dataset (frequencies of the products), exactly as required for significance testing. We formulate the hypothesis as follows:

- H_0: null hypothesis – the frequency Fr_X of a k-gram is not significant, and the k-gram does not have a sequential nature.
- H_1: alternative hypothesis – the frequency Fr_X of a k-gram is significant. It is unlikely that the frequency comes from the random i.i.d case.

After setting the significance level α, which corresponds to the probability of rejecting H_0 when it is true, we test the hypothesis by checking the following p_{value}:

$$p_{value} = Pr[B(T, Pr(X)) \geq Fr_X].$$

If $p_{value} \leq \alpha$, the null hypothesis H_0 is rejected, and the frequency of k-gram is considered to be statistically significant. After applying the hypothesis testing in Section 5, we demonstrate that our company-product time series have strong temporal correlation, rejecting therefore H_0.

3 Related work

Several approaches for company-product modeling have been proposed and aim at product recommendations. The first group of methods includes various co-clustering algorithms, for example the PaCo algorithm [13] and the OCuLaR algorithm for overlapping co-clustering [3]. These algorithms do not consider the sequential nature of data and are therefore not related to our work.

Of the sequential algorithms applicable to our problem , we can cite various 'heavy hitter approaches', such as association rules and conditional heavy hitters [10], [8]. These algorithms are able to model time series in a Markovian fashion to produce product predictions, but they are not able to build hidden structures of the IT install base of companies. Moreover, they do not provide high-dimensional product representations.

The third group of techniques comes from the NLP domain. One of the key tasks in NLP is language modeling. The goal is to learn representations for hierarchies of concepts, starting from words, phrases, and sentences, to more sophisticated concepts, such as documents and topics. Inspired by our initial results in modeling company-product data with Latent Dirichlet Allocation [9], we decided to consider other approaches from the NLP domain. In recent years, a lot of research has been devoted to the advancement and improvement of language processing methods, including Deep Neural Network (DNN) approaches. We assume that our company install base model consists of the following layers: a layer of companies, a layer of product categories and a hierarchy of latent structures inside the install base. Given this assumption, these company layers can be mapped to NLP concepts for application in DNN methods. Considering NLP terminology, we associate companies with documents and product categories with words. The vocabulary of words consists of the product categories in our scope. All companies that we consider in our analysis form the corpus of company documents. We further assume that products or product embeddings can be grouped into latent topics, which then construct specific and discriminative features $\mathscr{B}_i$ for each company c_i, $0 = 1, 2, ..., N - 1$.

Currently DNNs are the core of the state-of-the-art techniques for the tasks related to NLP. Mikolov *et al.* [7] [6] use a simplified architecture of neural networks that allow the use of very large training datasets and the building of accurate word embeddings in Euclidean space of high dimensionality in a very efficient manner. The word embeddings can afterwards be used directly for clustering without any transformation or aggregation as features. They can also be aggregated to represent a document as a vector in a smaller space using, for example, the Fisher Kernel Framework (probabilistic modeling of the corpus of documents using a mixture of Gaussians [5]) similarly as described by Clinchant *et al.* [2].

If labeled data is provided, classification can be done with a convolutional neural network, where both embedding vectors are refined and the classifier is learned similarly as in the approach proposed by Severyn *et al.* [11]. As our problem is unsupervised, we focus in the direction of unsupervised DNN, such as learning of word embeddings and generative text modeling using Recurrent Neural Networks. RNN achieve state-of-the-art performance on language modeling, which is what we are interested in, and on the tasks of speech recognition and machine translation [14]. This technique applies a distinct view on the data flow, using information not about independent instances but taking into account the sequential nature of data. Therefore, recurrent networks are sensitive to the past inputs and can adapt to them. State-of-the-art performance on language modeling task is achieved by RNN with Long Short-Term Memory (LSTM) units with the dropout regularization method [14]. We will apply this method to our company-product modeling using a time series input, A_i, and analyze its performance.

4 Solution Approach

In the general case of recommender system building, the goal is to learn the parameters of a scoring function $f(c_i, a_j)$. The function $f(\cdot)$ takes as input a company c_i, its current product time series A_i and a possible new product a_j, and provides the score of recommending the product a_j for company c_i. This scoring function can be linear, logistic, a multilevel neural network or can belong to another family of functions. The domain of values of a scoring function can be binary, real numbers or it can provide a probability that the offering will be successful in the future with a certain set of company and product parameters. If historical data about all the features of a company and its offerings is available, several time slices of data can be used to do supervised learning of the parameters of a scoring function. As in our case we do not yet have enough data from our internal sources for supervised learning, we focus on the methods of unsupervised learning. Once we have additional historical and labeled data in the future, we can use these representations of companies and products as an input to supervised learning techniques.

In the current setting, we can model and learn the semantic information about companies in terms of their IT install base, which is based on the fact that similar products should be close in the L-dimensional space, where L is the size of the product embeddings.

In this work, we consider two types of unsupervised models: n-gram modeling and RNN-based modeling. The modeling methods are evaluated using the measures of goodness of fit of a model.

Model adaptation and parameter estimation. For RNN we use various architectures as parameters of the model. We select the parameters by minimizing the perplexity level of a model. The average per-product perplexity is calculated on a test set using the product time series A_i with the total number of products n from the test set. Perplexity[3] shows how well the probability distribution, defined by a model, predicts testing data and is calculated as follows:

$$Perplexity = \exp^{-\frac{1}{n}\sum_{i=1}^{n}\ln P(a_i)},$$

where $P(\cdot)$ is the probability distribution that is induced by the model. The lower the perplexity, the better the model. The embeddings of products $\mathscr{B}$ are computed using the RNN models with the lowest perplexity. Instead of the original nominative time series A_i, a company now can be represented via product embeddings trained on RNN. We also use the perplexity to assess n-gram models.

5 Experimental evaluation

The experiments are done for 860k companies and 38 product categories[4]. The companies belong to 83 industries, such as "Health Services", "Agricultural Services" and others.

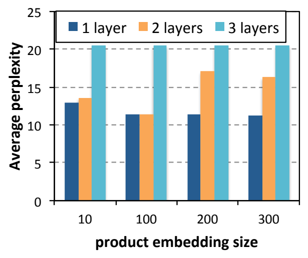
First, we estimate the percentage of statistically significant bi- and trigrams to check the sequential nature of our data. As described in Section 2, we use a significance level $\alpha = 0.05$. For $T = 4 \times 10^6$, $3650k$ bigrams and $3249k$ trigrams, we found that 69% of the bigrams and 43% of the trigrams have $p_{value} < \alpha$, therefore leading to the rejection of the null hypothesis. This means that these bi- and trigrams are statistically significant showing the clear sequential nature of our data.

Second, we estimate perplexity values of both the n-gram and the RNN model for our data. We use 70% of the initial corpus for training, 10% for parameter validation and 20% for model testing. As a baseline, we calculate the perplexities of 'naïve' initial times series representations, using 1-gram, bi- and trigram models. The resulting perplexities are correspondingly equal to 19.5, 16.4 and 15.5. In addition to the hypothesis testing result, we observed that longer contexts lead to better perplexity of n-gram models.

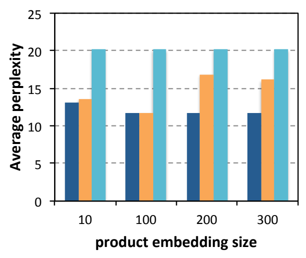
We use different architectures of RNN models by varying the number of hidden layers and the number of LSTM nodes, which is equivalent to the embedding size. We have tested 12 architectures of RNN with the number of layers $N_{layers} = \{1, 2, 3\}$ and the size of product embeddings $N_{embed} = \{10, 100, 200, 300\}$. We used the RNN model implementation from the 'tensorflow' package [1]. Training is done for 14 epochs over the training data. The resulting perplexity values for each RNN architecture are shown in Figure 1.

[3] We use the terms perplexity and average perplexity per product interchangeably.

[4] The full list of product categories is available at: `http://www.hgdata.com/Technologies-We-Track`.



(a) Average perplexity per product for training data.



(b) Average perplexity per product for testing data.

Fig. 1: Average perplexity of different RNN architectures.

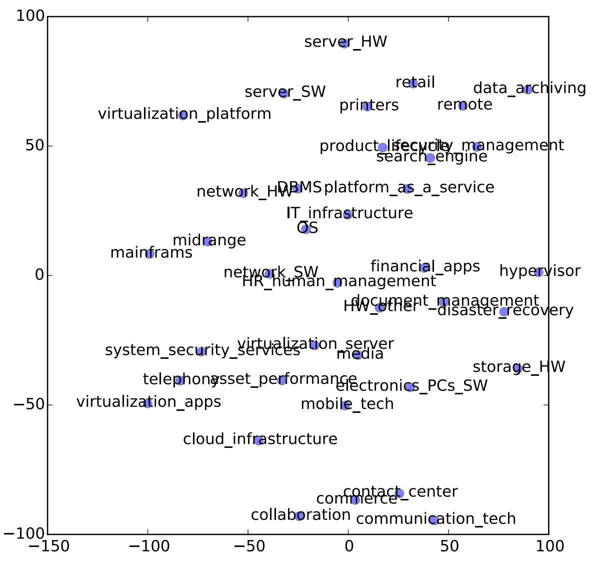


Fig. 2: Representaion of RNN-based product embeddings.

The lowest perplexity achieved throughout all the assessed RNN models is 11.6 for the test set, which corresponds to 1 hidden layer and an embedding size of 200. The perplexity is better than the baseline values. The RNN-based representations of products learned by the best model are represented in Figure 2 using t-SNE [12] 2D projections[5].

In this figure, we see certain intuitive "clusters", such as, for example, "collaboration - communication technologies - commerce and contact center" or "mainframes - midrange". As future work, the quality of this product representation will be evaluated in the context of a recommendation system.

[5] We shortened the original names of product categories for the sake of better visualization. 'HW' stands for hardware, 'SW' for software, 'OS' for operating systems.

6 Conclusions

We have demonstrated that unsupervised modeling techniques borrowed from the NLP domain are able to capture intrinsic hidden install base structures in our company-product time series. The predictive capabilities are demonstrated by the resulting perplexity. To the best of our knowledge, this is the first time such techniques are used in this context. We have evaluated different model architectures and demonstrated that RNN with 1 hidden layer and an embedding size of 200 fits our data best.

As future work, we will assess other deep neural network architectures starting from lower levels of product descriptions. This presents the challenges of a much larger data set, which, in turn, is useful for training more sophisticated models. Another important work is to validate the results based on the outcome of a marketing recommendation tool. We will also compare the efficiency of RNN-based representations with the state of the art recommender system approaches, such as matrix factorization algorithms.

References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
2. S. Clinchant and F. Perronnin. Aggregating continuous word embeddings for information retrieval. *ACL 2013*, page 100, 2013.
3. R. Heckel and M. Vlachos. Interpretable recommendations via overlapping co-clusters. *ArXiv e-prints*, Apr. 2016.
4. HG Data Company. `http://www.hgdata.com`.
5. T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.
6. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
7. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119, 2013.
8. K. Mirylenka, G. Cormode, T. Palpanas, and D. Srivastava. Conditional heavy hitters: detecting interesting correlations in data streams. *The VLDB Journal*, 24(3):395–414, 2015.
9. K. Mirylenka, C. Miksovic, and P. Scotton. Applicability of latent dirichlet allocation for company modeling. In *Industrial Conference on Data Mining (ICDM'2016)*, 2016.
10. K. Mirylenka, T. Palpanas, G. Cormode, and D. Srivastava. Finding interesting correlations with conditional heavy hitters. In *IEEE 29th International Conference on Data Engineering (ICDE)*, pages 1069–1080, 2013.
11. A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM, 2015.
12. L. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
13. M. Vlachos, F. Fusco, C. Mavroforakis, A. Kyrillidis, and V. G. Vassiliadis. Improving co-cluster quality with application to product recommendations. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 679–688. ACM, 2014.
14. W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.