*Industry and Applications Paper*

# Hidden Layer Models for Company Representations and Product Recommendations

[Application Paper]

Katsiaryna Mirylenka
IBM Research – Zurich
Switzerland
kmi@zurich.ibm.com

Paolo Scotton
IBM Research – Zurich
Switzerland
psc@zurich.ibm.com

Christoph Miksovic
IBM Research – Zurich
Switzerland
cmi@zurich.ibm.com

Jeff Dillon
IBM Canada
Markham, Canada
jdillon@ca.ibm.com

## ABSTRACT

An increasing amount of marketing intelligence data is becoming available today. This includes data that describes information technology (IT) inventories, i.e. IT products purchased by companies. It is advantageous for hardware and software service providers to analyze this data and build recommender systems to propose new products for client companies. Real-time recommendations are usually done based on matrix factorization methods or association rules. In this work we study the applicability of generative models to the recommendation problem. We focus on models that are able to reveal latent connections between companies and deployed IT products and build discriminative features of the IT structure of a company. Additionally generative models of company-product data are of interest for service providers for efficient company comparison, application of similar marketing strategies towards the groups of similar companies. In this work, we first formalize the notion of a company and its IT install base. Then, we estimate various generative models that are able to reveal hidden structures in data. These are mainly topic and language modeling techniques emerging in natural language processing to the task of company-product modeling and sequential models that are widely used for product recommendations. More precisely, the analysis is done using (a) Latent Dirichlet Allocation (LDA) with the products in a company are treated as a set, (b) $n$-gram models or sequential association rules and (c) Recurrent Neural Networks (RNN), when the time of product appearance is taken into account. The techniques are used for a corpus consisting of 860k companies. The results of the study demonstrate that simpler generative models with lower number of parameters, such as LDA, fit company-product data better and are more beneficial for company IT install base modeling both in terms of goodness of fit of the model and recommendation quality.

## 1 INTRODUCTION

Information technology (IT) install base[1] data is provided by specialized companies that carefully maintain its quality in terms of confidence of IT products' presence and accuracy of timestamps of their appearance. This kind of data has already been exploited for industrial applications and proved to be extremely useful to get market insights in several contexts such as, for example, new market development or white space determination. Data usually refers to companies. For a specific company, different types of information are provided, for example, insights about the internal structure, what the company buys, etc. IT install base data is a specific type of marketing intelligence data that provides, for a set of companies, knowledge about the type of IT equipment they own and how this equipment is distributed across their physical locations.

Install base information is particularly useful when addressing white spaces. If a provider of hardware services tries to acquire new customers, install base information will illuminate the insights about the business potential for a particular product domain. More interestingly, when combined with data about established customers, install base information can be used to identify companies that are similar to existing clients and therefore have a high potential of becoming new customers by acquiring certain sets of products.

In this paper, we concentrate on the problem of modeling company-product relations in order to (1) identify similar companies and (2) build product recommendations. Besides the computational complexity of the similarity search problem due to the large number of companies and product types, another challenge is that a naïve comparison of the individual product types owned by companies turns out to be biased towards product types that are common to a large number of companies. Therefore, after reviewing the related work, we focus on two approaches that consider the hierarchical similarity of objects (companies) based on contextual proximity of their features (product types). This happens to be an assumption in the field of Natural Language Processing (NLP). In our case, the similarity on the lower level is defined via the co-occurrence of the product types in a company. Higher levels correspond to hidden structures in the install base and the representation of a company itself.

To get the best model representations of products and then of companies, we estimate and evaluate various types of generative models. We choose the best model in terms of goodness of fit and predictive power. This model is used to extract product and company representations which are then applied for similarity search queries and marketing recommendations. In particular, we compare the performance achieved by these two modeling techniques: Latent Dirichlet Allocation (LDA) and Recurrent Neural Networks (RNN). The products belonging to a company compose a multilevel structured representation of the IT install base. Such

[1] Install base refers to the IT inventory of a company.

products are used as the basis of company similarity evaluation. The LDA and RNN techniques are beneficial for the IT install base modeling as they are able to reveal hidden hierarchies. RNN is preferred over other neural network architectures as it is able to take into account time-correlations of product time series. Besides, both techniques are among the best performers in the corresponding domains. LDA produces interpretable parameters, while the embeddings obtained by RNN modeling lack the interpretability. Uninterpretable parameters is a significant drawback for business applications, which can be tolerated if these models would perform much better.

The main contributions of this work are the following:

(1) We formalize the problem of modeling the IT install base of companies for various types of input data, such that the state-of-the-art unsupervised techniques from NLP domain can be applied. First, we formalize company-product data treating products in a company as a set and use corresponding non-sequential models for this case. Second, we treat the product data as sequences according to the time of their appearance. The models are then compared using their data fitting quality.
(2) The applicability of topic modeling via LDA and language modeling via RNN is assessed for our data. We demonstrate that LDA with 2, 3 and 4 latent topics fits the data best and, additionally, provides the most discriminative company features for the task of company clustering.
(3) We assess the practical applicability of the LDA and RNN models to an industrial product recommender and compare them with sequential association-rule and matrix factorization approaches.
(4) After solving various data integration challenges, we apply the output of the best performing model, enriched with internal company-product data, to a recommendation tool that searches for similar companies and makes recommendations.

## 2 PRELIMINARIES AND PROBLEM FORMALIZATION

The goal of this paper is to develop a recommendation system that also allows companies to be compared based on their install base. This comparison will be used to generate sets of similar companies and will allow possible business developments to be identified in real time. As the system in development is to be used by offering managers, the interpretability of the results is a crucial requirement to be able to justify the outcome. In consequence, models with interpretable parameters bring an advantage in our settings.

As a source of marketing intelligence data, we use information provided by HG Data Company, Inc. [12]. HG Data Company is building and maintaining a comprehensive database with competitive intelligence about deployed technologies. Essentially, for each company assessed, this database provides the following information: the type of IT products available at each site of the company without specifying the quantities and product model details, some indication about the confidence of the information provided, and dates of the first as well as the most recent successful confirmation of product presence.

Product descriptions are organized in a hierarchical fashion. This hierarchy contains four levels. At the top, we find the *vendors*. For each vendor, there is a list of *category parents* giving a high-level grouping of the product types, for example, "Data Center Solution" or "Hardware (Basic)". Each *category parent* contains a list of *categories*, which are finer-grain groups. Examples of categories are "Printers" or "Midrange Computers". Finally, each category contains the *product types* available for the vendor considered.

Our aim is to develop a sales recommendation application. Given a customer, potential customers with similar IT install base are provided as input to our solution, which then combines this information with internal data. To align the product description between our proprietary data and the result of the company similarity evaluation, we have focused on the category layer. Unfortunately, the product types do not contain a certain product details, thus, do not allow us to link the products with the lowest level of product descriptions in our proprietary data.Therefore for each company we consider the product categories[2] associated with the company, independently of the vendor. In our version of the HG Data Company database, there are 91 distinct categories. Out of those categories, we decided, because of the nature of our application, to restrict our study to hardware and low-level hardware management software categories, thus, using 38 distinct categories.

To link data from the HG Data Company database to our internal data, we solved integration and cleaning issues. In the HG Data Company database, companies are identified by their D-U-N-S® numbers. This number is a unique 9-digit identifier, assigned by Dun & Bradstreet, Inc. [8] to each business location. Therefore, each company entity, such as branches, subsidiaries and headquarters, has an individual D-U-N-S® number, and the set of all D-U-N-S® numbers associated with a company is organized hierarchically.

After we had chosen the product categories or company attributes and aggregated the subsidiaries of a company, we created our corpus for model training which is a binary company-product matrix. As we do not have information about product quantities, we consider only binary values in the company-product matrix, where 1 means that the product belongs to the install base of the company and 0 means that it does not.

More formally, let us 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 given set of products $A_i$ in its install base belonging to $k$ categories, which can also be called attributes. This set of attributes is included in the set of all possible attributes $A = \{a_0, \ldots, a_{M-1}\}$ containing $M$ elements[3]. That is

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

The attributes might be sorted by the time of their appearance in the IT install base of a company and treated as data series. We denote the sorted version of $A_i$ as $A_i^S$. The information about the attributes or products from $A_i$ can be re-written using vectors $\mathscr{A}_i$ instead of sets of products:

$$\forall c_i \in C\ ;\ \ c_i \longmapsto \mathscr{A}_i\ ,\ \ \dim(\mathscr{A}_i) = M\ , \tag{2}$$

$$\mathscr{A}_i = \left[\mathbb{1}_{a_0 \in A_i}, \ldots, \mathbb{1}_{a_{M-1} \in A_i}\right]. \tag{3}$$

A naïve approach to compare companies is to calculate the distance between their initial attributes $\mathscr{A}$ or $A^S$. In Section 3, we discuss why such an approach leads to results that are not sufficiently meaningful. The initial attribute space does not represent companies in a discriminative manner because the distance

[2] In the remainder of the paper we use the terms products and product categories interchangeably, meaning product categories.
[3] In our application $M = 38$.

between companies is affected too strongly by the most popular attributes. To overcome this problem, we introduce a new space of company features that better represents the IT install base:

$$\forall c_i \in C\ ;\ \ c_i \longmapsto \mathscr{B}_i \in \mathbb{R}^L\ ,\ \ L < M. \tag{4}$$

Considering the new company features $\mathscr{B}$, it is possible to express the distance between two companies as a classical distance between two vectors:

$$\forall c_i\ ,\ \ c_j \in C\ ;\ \ dist(c_i, c_j) = d(\mathscr{B}_i, \mathscr{B}_j), \tag{5}$$

where $d(.,.)$ is any vector distance, e.g., euclidean or cosine distance.

One of the goals of this paper is to automatically discover the most representative features of a company $\mathscr{B}$, based on initial company attributes $A^S$ or attribute vectors $\mathscr{A}$. The features should be representative in terms of goodness of fit of the generative model of company-product data and in terms of quality of similarity clusters of companies. The methods of feature learning for companies are discussed in Section 4. Learned company and product features are, then, used for product recommendations.

## 3 RELATED WORK

In this section, we consider the applicability of existing methods for building discriminative company representations, namely, the feature vectors $\mathscr{B}_i$ for each company $c_i$. $\mathscr{B}_i$ are built using product vectors of a company, $A_i$, or a sequence of products per company, $A_i^S$. This representations should capture the hidden structures in the company-product data.

### 3.1 Co-Clustering

Co-clustering or bi-clustering is a technique used for building two-dimensional groups of objects that are represented by a matrix. The matrix is clustered along the rows and the columns. The principle of co-clustering was first introduced by Hartigan in 1972 [10]. Since then, several algorithms have been proposed mainly aimed at product recommendations.

The PaCo algorithm [27] and the OCuLaR algorithm for co-clustering with overlapping [11] are most closely related to our problem. The main issue with these approaches is that they are built on initial company representations $\mathscr{A}_i$, which may be not the best features to distinguish IT install bases of companies. When we applied the PaCo algorithm as well as spectral co-clustering to a small sample of around 500 companies that belong to a healthcare industry, we could not generate meaningful co-clusters. The only co-cluster generated contained overall popular products. Our first attempt of using LDA for this small subset of companies and products [18] provided us with much better results and inspired us to continue our search of company features in this direction. We believe that modified product features could improve the quality of co-clusters. As we also show in Section 5, raw initial company-product representations $\mathscr{A}_i$ neither describe the generative model of our data well nor perform well for the task of company clustering. In addition, given the large number of companies to analyze (on the order of a million), co-clustering results are difficult to interpret because the intuitive visual way to consume co-cluster results is not possible in this case.

### 3.2 Pattern mining

Another possibility is to explore product install base modeling using the approaches of the frequent pattern search from the time series domain. One family of such approaches is Association Rule mining [2], which is partially time agnostic. It was demonstrated in [6], [7] that taking into account the sequential nature of data is beneficial for time series tasks, such as similarity matching and nearest neighbor search. This inspired another line of techniques that are based on estimating Markov Chain models via real-time algorithms for 'conditional heavy hitters' discovery [20], [17]. However, the mined patterns are not able to represent the hidden structures of the IT install base of companies, though we use them to compare more advanced methods with.

### 3.3 Applicability of NLP Concepts

In the past decade, much attention in the literature was given to modeling techniques related to NLP. In this subsection, we discuss their applicability to the problem of modeling install bases of companies. 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, which are then organized into more sophisticated concepts, such as documents and topics. In recent years, a lot of research has been devoted to the advancement and improvement of topic modeling and language processing methods, including, among others, LDA and Deep Neural Network (DNN) approaches. We assume that the generative model of our company install bases is similar to the NLP models. The product-company world 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 the LDA and DNN methods. Considering NLP terminology, we associate companies with documents and products with words. 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 $c_i$, $0 = 1, 2, ..., N - 1$.

### 3.4 Deep Neural Networks and Product Embeddings

Currently DNNs are the core of the-state-of-the-art techniques for the tasks related to NLP. Mikolov *et al.* [16] [15] use a simplified architecture of neural networks that allows them to use very large training datasets and to build accurate word embeddings in Euclidean space of high dimensionality very efficiently. The word embeddings can afterwards be used directly without any transformation or aggregation as features for clustering. They also can 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 [14]) similarly as described in the work [5].

The larger the training set (hundreds of millions) and the larger the dimensionality of the word embedding, the better is the representation, although after a certain point the improvement is no longer significant. This is partially due to the large size of the vocabulary that typically needs to be learned, namely, 600K. As in our case the number of products is much smaller, there is a chance that we will learn good embeddings from tens of thousands of companies.

State-of-the-art performance on language modeling task is achieved by RNN family of models and specifically by RNN with Long Short-Term Memory (LSTM) units with the dropout regularization method [29]. This technique applies a distinct view

on the information 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. Other RNN realization, such as Gated Recurrent Unit (GRU) [4] which is a simpler version of LSTMs, has recently gathered popularity, but study [9] empirically demonstrates the the performance of GRUs and other RNN architectures can be better for some datasets, but do not outperform LSTM in general. We will apply LSTM architecture to our company-product modeling using a time series input, $A_i$, and analyze its performance both in terms of goodness of fit and as a recommender of future IT products of companies.

In work [19], we have demonstrated that LSTMs are applicable to the task of modeling company-product time series. We used RNNs with Long Short-Term Memory (LSTM) units with the dropout regularization method [29]. Prior to applying RNNs in our work [19] we demonstrated that company-product time series are of clear sequential nature. This was done using statistical hypothesis testing and was based on the fact that the frequency of the i.i.d. time series observations has binomial distribution. In this work, we compare accuracy of the RNN modeling, where product time series are taken into account, with the performance of LDA, where products in a company are treated independently, without their timestamps. We also assess both methods for the recommendation task, comparing their results with the association rule-based recommender.

### 3.5 Latent Dirichlet Allocation

Blei *et al.* [3] introduce LDA, which is a generative probabilistic model for collections of discrete data, such as corpora of documents (or company-product vectors in our case) that is also used for collaborative filtering. Each item, such as a document or a company, is modeled as a finite mixture of an underlying set of topics or hidden groups. The topic probabilities for an item provide an explicit representation of a document. In parallel, word embeddings are also trained. The embeddings represent relatedness of words in the space of document topics.

The number of latent topics is a user-defined parameter. It can be chosen using measures of the goodness of fit of the LDA model, such as, a total log-loss for a testing set of documents or as the average perplexity of how well each single word is modeled. In our case of company-product modeling, LDA learns both product embeddings and company representations $\mathscr{B}_i$ in the vector space that is the size of the number of latent topics. The main advantage of LDA over RNN and other topic modeling techniques such as Latent Semantic Indexing [13] is that LDA-learned features are easy to interpret. This fact is important for adopting those techniques in marketing environment.

## 4 MODELING APPROACH

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 and, then, similar companies should be close in the $L$-dimensional space, where $L < M$. In this case, the recommendations are extracted using the notion of similarity between the companies. Recommendations are based on the similarity calculated given the dataset from HG Data Company. The gaps in possible product offerings are extracted from our internal databases for similar companies. The strength of the recommendation is in this case measured via the strength of the company similarity.

We compare two types of unsupervised models: non-sequential modeling (LDA-based), when products are considered independently ($\mathscr{A}$ company features are used as input), and sequential modeling (LSTM-based), when we take into account the order of product appearance in the HG Data Company database via $A^S$ input. Both modeling techniques capture the hidden structures in the company-product data and produce features (representations) for products and companies. We assess the quality of the following company-product representations:

(1) Naïve representation: binary or Term Frequency-Inverse Document Frequency (TF-IDF) [22] vector of products. In our case, TF-IDF can be also reformulated as product frequency-inverse company frequency.
(2) RNN-based representation: embedding (vector) that shows the position of a company in an $L$-dimensional vector space. The position depends on the contextual similarity of products of a company.
(3) LDA-based representation: vector of real numbers that shows the probability that a company belongs to an LDA topic.

The modeling methods are evaluated using the measures of goodness of fit of a model, and, additionally, the representations are evaluated for the company clustering task. When the proper representation is chosen, we can find the top-$k$ similar companies based on HG Data Company data. The models are also evaluated for the recommendation task.

### 4.1 Model adaptation and parameter estimation

We train LDA both on initial binary company-product representations[4] and TF-IDF representations. The type of data representation is considered as one of the parameters for LDA training. Although LDA intrinsically models data to give more weight to the most representative features, we verify whether the model improves if TF-IDF representation is given as an input. Another crucial parameter of LDA modeling is the number of latent topics; this number is chosen using goodness of fit measures.

For RNN, we used various architectures as modeling parameters. More details about LSTM modeling can be found in [19]. We select the parameters of LDA and LSTM by minimizing the perplexity level of a model. The average perplexity per product is calculated on a test set using $\mathscr{A}_i$ or $A_i^S$ company representation, with the total number of products being $n$. Perplexity[5] shows how well the probability distribution defined by a model (for example, LDA or LSTM) 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 induced by a model. The lower the perplexity, the better the model. The best features of a company $\mathscr{B}$ are computed using the models with the lowest perplexity. Instead of the original binary vectors $\mathscr{A}$ or binary sets of products $A^S$, a company is represented via the vector of latent LDA topics or company embeddings trained on RNN.

[4]Initial binary representation can also be called 'Bag Of Words' (BOW) representation in terms of NLP theory.
[5]We use the terms perplexity and average perplexity per product interchangeably.

## 4.2 Validation of Company Representations using Clustering

To see how extracted features perform in comparison with initial binary features, or initial TF-IDF features, we assess the quality of learned representations for a clustering task.

As measure of clustering quality, we use silhouette scores[6]. The silhouette score is calculated as the ratio of intraclass and interclass distances. The higher the score, the better the clusters are separated from one another. We choose an appropriate model depending on the silhouette score value for the desired number of clusters. We expect that the model with the highest silhouette scores is also the best according to the perplexity results, as both validation measures favor the most descriptive and representative features.

## 4.3 Recommendation Capabilities of Generative Models

In addition to the perplexity evaluation, we also check the recommendation capabilities of LDA, LSTMs and n-gram based models. For the marketing recommendation scenario, it is essential to provide an outlook over a relatively long period of time; typically the span of interest $r$ ranges from 6 to 24 months. To evaluate this capability, we assess the model recommender for a sliding window $W_r$ that covers $r$ months. This time window slides with a granularity of two month in order to accumulate significant changes in the install base of a company. At each iteration, the recommender provides a series of observations *i.e.* the probability of a given product appearing within a particular window $W_r$. In order to access the statistically significant accuracy, we gather the accuracy information for a number of sliding windows. The cardinality of the set of accuracy observations is equal to the number of sliding windows and is denoted as $l$. All the previous information that happened before the start of a sliding window is used for model training.

To obtain a recommendation for a particular product, we assess the conditional probability of seeing that product appearing given the time series of products that a company acquired so far. The probability is obtained as the output of a model. If for a product $p_i$ the probability of the generative model $M$, $Pr(p_i|M, p_{i-1}, p_{i-2}, ...)$ exceeds a threshold $\phi$ we assume that the product $p_i$ should be recommended to a given company. Products $p_{i-1}, p_{i-2}, ...$ represent previous products acquired by that company. As the optimal probability threshold $\phi$ is not known in advance we treat it as a parameter of the validation of the model recommender. For each $\phi$ and for each company time series we estimate the average precision and recall measures for the company time series and all the sliding windows $l$. Recall shows how many of the true products that company buys in the future are retrieved by the recommender. Precision represents the ratio of the retrieved products that are in the true future product set.

# 5 EXPERIMENTAL EVALUATION

The experiments are done for 860k aggregated companies and 38 product categories[7]. Company aggregation is performed using domestic D-U-N-S® numbers, that is, all company sites in one country are aggregated. Products are consequently aggregated into a set containing all products available in all sites of a company. The companies belong to 83 industries, such as "Health Services", "Agricultural Services", etc., which are encoded with the SIC2 codes.[8]

First, we estimate the perplexity of initial company representations $\mathscr{A}$. This is equivalent to the perplexity of the unigram 'bag of words' model. The perplexity is equal to 19.5. The perplexities of bi- and tri-gram models are also reported in [19]. Their value is not lower than 15.5, which will be the evaluation baseline.

**LSTM.** For LSTM, we used $A^S$ company representations, where products are sorted according to the date of their first appearance in a company. LSTM is applicable for sequential data. The sequential nature of our data was checked in work [19], where we demonstrated that 69% of the bigrams and 43% of the trigrams have frequencies that are statistically significantly higher than in the case of independent identically distributed products. This demonstrates that the time dependencies among the products are indeed strong. The hypothesis testing was based on the binomial distribution of frequencies of $n$-grams.

We used 12 different architectures of LSTM model by varying the number of hidden layers ( $N_{layers} = \{1, 2, 3\}$) and the number of nodes in the layers ($N_{nodes} = \{10, 100, 200, 300\}$). We used 70% of the initial corpus for training, 10% for parameter validation and 20% for model testing. We used the LSTM model implementation of the 'tensorflow' package [1]. Training was done for 14 epochs over the training data. The resulting perplexity values for each RNN architecture are given in Figure 1.

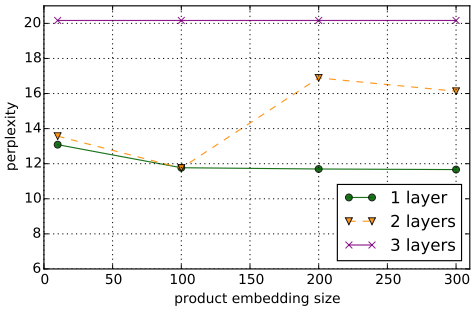


**Figure 1: LSTM average perplexity per product for test data.**

As can be seen from the results, the lowest (best) perplexity achieved by LSTM model is 11.6 for the test set, which corresponds to 1 hidden layer and 200 nodes per layer. The number of nodes per layer corresponds to the product embedding size. LSTM also learned meaningful representations of IT products, that are illustrated and discussed in [19].

**LDA.** In the case of LDA, we need to set the number of latent topics. Altough LDA takes into account the representativeness of words (products in our case), we consider both raw binary representations and TF-IDF representations as input for the model. The division of the corpus into training and test datasets is done in the same way as for RNN modeling. The LDA implementation used is from the *gensim* package [23]. The perplexity curves of LDA for both inputs are shown in Figure 2.

[6]For the experiments, we use the silhouette score implementation available in Python programming language (Python Software Foundation, https://www.python.org) package *sklearn*[21]

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

[8]Standard Industrial Classification (SIC) is a taxonomy established by the US Government to classify industries. The full list of SIC encoded industries is available at: http://siccode.com/.

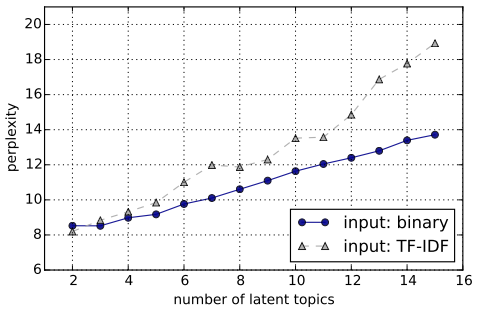


**Figure 2: Average perplexity plots measured on test data for LDA models.**

It appears that the perplexity of raw binary inputs is better than that of TF-IDF pre-processed input. The leads to the conclusion that LDA indeed is able to assign higher weights to the most representative products and that therefore, no additional pre-processing is needed. Moreover, lower numbers of hidden topics, namely. 2, 3 and 4 lead to lower perplexity values, which vary from 8.5 to 8.9.

The minimum achieved perplexities for each method with the ranking placing the best methods first are shown in Table 1.

**Table 1: Minimum perplexities achieved by each method varying the parameter settings.**

| | **Method Name** | Min. Perplexity |
|---|---|---|
| 1 | LDA | 8.5 |
| 2 | LSTM | 11.6 |
| 3 | $N$-grams | 15.5 |
| 4 | Unigram 'bag of words' | 19.5 |

**Lessons learned.** We conclude that, for our company-product data, LDA models perform better than LSTM models. Note that LDA models do not take into account the time component of the company-product space, whereas RNN is built over time ordered sequences of products. We assume that more sequential training data might be needed to build more accurate LSTM models as they have more parameters than LDA models. Indeed, the number of parameters to estimate in LDA models is equal to $n_t + n_t * M$ [3], where $n_t$ is the number of latent topics in the LDA model and $M$ is the vocabulary size of products. In the case of four latent topics, we have 156 parameters to estimate. The number of parameters to estimate for LSTM is dominated by $n_c * (4 * n_c + n_o)$ factor [24], where $n_c$ is the total number of memory cells and $n_o$ is the number of output units. In the case of one the best performing LSTM settings in our deployment with 100, the number of parameters is lower bounded by $100(4 * 100 + 100) = 50000$.

Another hypothesis is that the distribution and properties of hidden structures in company-product data correspond better to LDA modeling assumptions. The fact that LSTM with only one hidden layer fits the data the best out of other LSTM architectures supports this hypothesis.

## 5.1 Recommendation accuracy

To assess the recommendation accuracy we follow the methodology described in Section 4.3. The time span of the product time series in our deployment range from 1990 till the end of January, 2016. We use a trained model (LDA, LSTM or $n$-gram based) to predict products within a sliding window $W_r$ of 12 month starting from January,1 2013, thus, $r = 12$. The window slides by two month. This way we obtain 13 accuracy observations. The first recommendation window starts on January 1, 2013 and finishes by January 1, 2014 and the last recommendation window starts on January 1, 2015 and finished on January 1, 2016.

The precision and the recall of the LDA3 is compared to the LSTM-based recommender and $n$-gram recommender based on exact Conditional Heavy Hitters [17], that is another data series technique capable of capturing time correlations in the data and predicting future states of the system. The exact Conditional Heavy Hitters are also exact time-dependent association rules. Based on the experiments to validate temporal correlations in product time series described earlier, the depth of the context for Conditional Heavy Hitters (CHH) is chosen to be 2. Thus, we study the dependencies on the previous products up to the second order.

The plot with the average Precision and Recall values, for different values $\phi$ of conditional probabilities that are used as a recommendation threshold is shown in Figure 3.

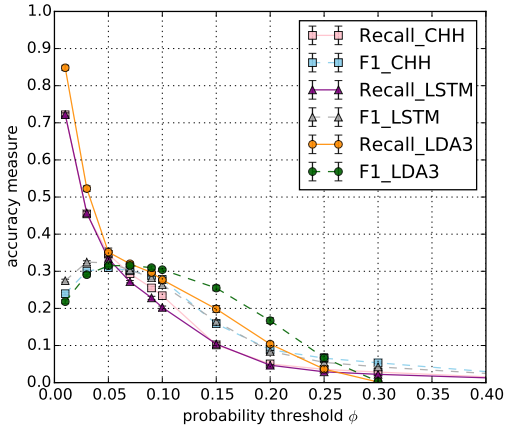


**Figure 3: Recall and F1-score with the corresponding confidence intervals for the recommenders based on LDA, LSTMs and Conditional Heavy Hitters (CHHs).**

The average numbers of retrieved and correctly retrieved products by each method with the confidence intervals are shown in Figure 4.

We can infer (Figure 4) that $\phi$ should be smaller or equal to 0.2, as for higher values the numbers of products retrieved by the methods are too low. The accuracy results demonstrate that the recall of a recommender based on LDA model is consistently higher than the recall of LSTM and CHH-based recommenders for the appropriate values of $\phi$. The F1-score is also higher for large range of $\phi$. For the values of $\phi \geq 0.25$ the recall values are not statistically significantly different as their confidence intervals intersect. Also we note that beyond the probability threshold $\phi = 0.5$ LDA and CHH did not produce any recommendations, thus, precision values are not defined for this points and recall values are equal to zero.

The plots on Figures 3 and 4 demonstrate also that the recall LSTM and CHH is similar as they retrieve similar number of true future products of a company, but CHH-based recommender

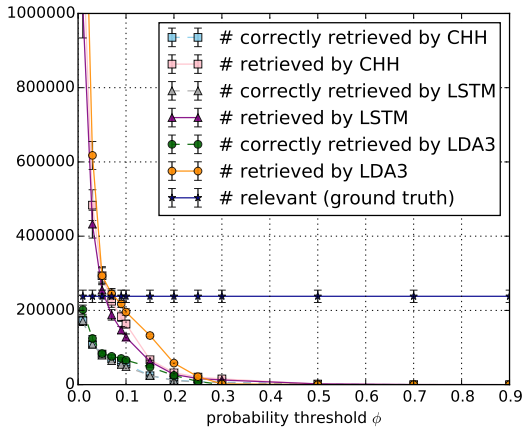


**Figure 4: The average number of retrieved, correctly retrieved and relevant products for LDA, LSTM and CHH with the confidence intervals.**

tends to produce more false positives, i.e. it recommends more products that should have not been recommended. This causes significant differences in precision.

The task of product recommendations is very hard as it is quite difficult to capture all the underlying processes that influence the choice of future models. This is witnessed by the fact that so far, the best models we tried are only able to reach the values of precision and recall around 0.25 - 0.43 for $\phi \in [0.05, 0.15]$. This means that they are able to capture a true generative model to some extend. The random generator that produced a product recommendation with a uniform probability = $1/38 \approx 0.026$ retrieved all the products for the thresholds $\phi <= 0.026$ and on average close to zero of the correct products for higher thresholds. The differences between the accuracy measures for LDA and LSTM, LDA and CHH-based methods are statistically significant for most of $\phi$ values as the corresponding 95% confidence intervals do not intersect. We have also assessed the accuracies LDA-based recommender with two and four latent topics, their performance has been very similar to the results of LDA with the three latent topics which has been described above. As a future work we will study the influence of the sliding window size on the recommendation accuracy.

## 5.2 Comparison with Matrix Factorization

We also compared the hidden layer methods with one of the best-performing matrix factorization techniques, Bayesian Probabilistic Matrix Factorization (BPMF), which was introduced in work [25] and implemented in [28]. As the system requires rankings as an input, we use a ranking transformation of the training and test data described in the previous section. This means that if a company has product $x$, its ranking is equal to 1, otherwise, its ranking is equal to 0.

The BPMF recommendation results that we obtained for the product recommendations in our deployment were similar to the initial results for co-clustering that we reported in Section 3.1. In the majority of the cases, recommendations produced by BPMF for a given company include all possible products. The distribution of the BPMF recommendation scores can be seen in Figure 5. This is due to the fact that BPMF, and matrix factorization methods in general, was developed for sparse and imbalanced datasets. The data in our deployment is relatively dense, and cannot be reasonably approximated by a low-rank matrix, which is the basis of BPMF.

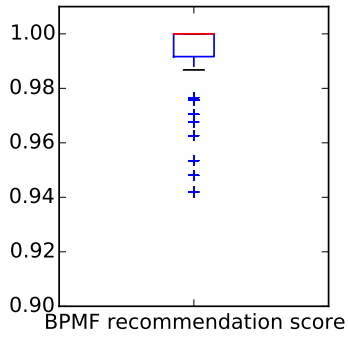


**Figure 5: Boxplot of BPMF recommendation score values.**

The precision, recall and F1-score values depending on varying BPMF recommendation score values (in the interval $[0.9, 1.0]$) are shown in Figure 6.

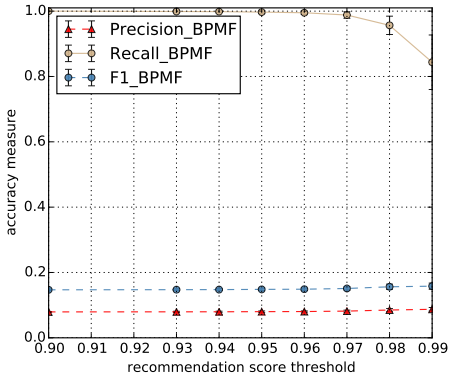


**Figure 6: Accuracy values of Bayesian Probabilistic Matrix Factorization method.**

All the scores for the thresholds of recommendation score that are lower than 0.94 are the same as for the values 0.91, 0.93 and 0.94. This means that almost for all of the threshold values the full set of available products is recommended regardless the previous products purchased by a company. Thus, BPFM does not produce meaningful recommendation results and does not provide us with effective features for company-product modeling. In contrast, additional feature search for products, as done in LDA and LSTM, leads to high-quality recommendations.

## 5.3 Company clustering.

As LDA is the best-performing model based on the analysis of perplexity and recommendation accuracy, we will now study the suitability of LDA-learned features for clustering. For this purpose, we build silhouette curves for the features obtained with the best-performing LDA configurations, that is, with the number of topics equal to 2, 3 and 4. We then compare these results with silhouette curves that are built on i) raw binary company-product representations, ii) raw TF-IDF company representations and iii)

LDA representations with TF-IDF input for 2 and 4 hidden topics. The results are shown in Figure 7.

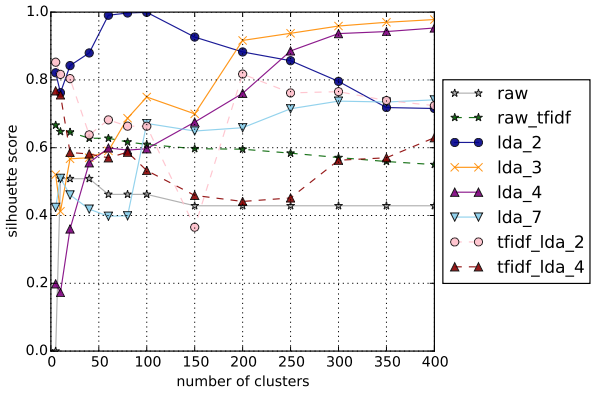


**Figure 7: Silhouette curves.**

Note that the higher silhouette score, the better the clusters of companies are separated. A higher score means that the distances between companies within one cluster is much lower than the distances between companies in different clusters. In Figure 7, we can see that the initial binary representations of companies are not very discriminative (blue line with stars) as the silhouette score is the lowest for almost all number of clusters. Clustering on the initial representation with TF-IDF transformation performs better then clustering on the initial binary representation, as the silhouette curve is higher and around 0.6 for a varying number of clusters. Clustering on LDA-generated representations with TF-IDF input (tfidf_lda_2 and tfidf_lda_4) performs better than clustering on raw TF-IDF, especially, when two latent topics are used. Company representation associated with the best silhouette curves are generated by LDA with raw binary inputs for the number of latent topics equal to 2, 3 and 4. This result is in accordance with the perplexity outcomes. This means that LDA with these numbers of latent topics represents the install base of the companies the best.

We notice that silhouette scores for lower numbers of clusters from 5 to 200 are higher for LDA representations with lower numbers of topics, i.e., 2 topics, whereas higher numbers of topics (3 or 4) discriminate larger numbers of clusters better.

The t-SNE [26] 2D projections for the product embeddings based on LDA3 and LDA4 are shown in Figures 8, 9. The original names of the product categories are shortened for better visualization, e.g., 'SW' and 'OS' stand for software and operating systems.

It appears that the main products that construct a topic produce clusters of products. It is also interesting to see that most of the hardware products are close to each other for both the LDA3 and the LDA4 representation. These are 'server_HW', 'storage_HW', 'HW_other'. Similarly, software products tend to appear together, for example, 'commerce', 'media', 'collaboration' 'product_lifecycle', 'electronics PCs SW' and 'retail'. Thus also the semantic proximity of the products is captured by LDA models.

**Lessons learned.** The good quality of company clusters and the meaningful representation of products discussed above mean that the LDA method is able to automatically infer representative features both for companies and products in our deployment.

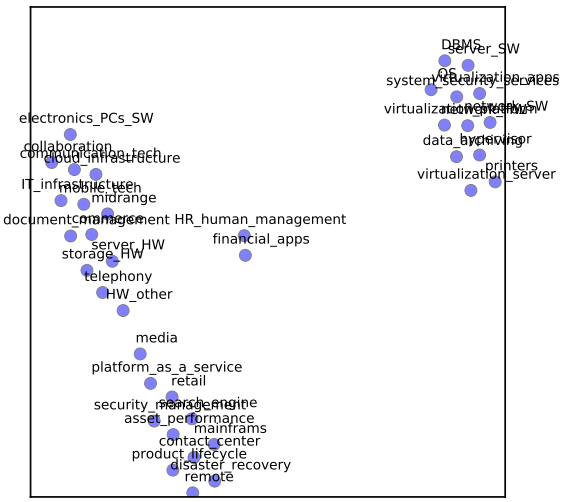


**Figure 8: LDA3 product embeddings.**

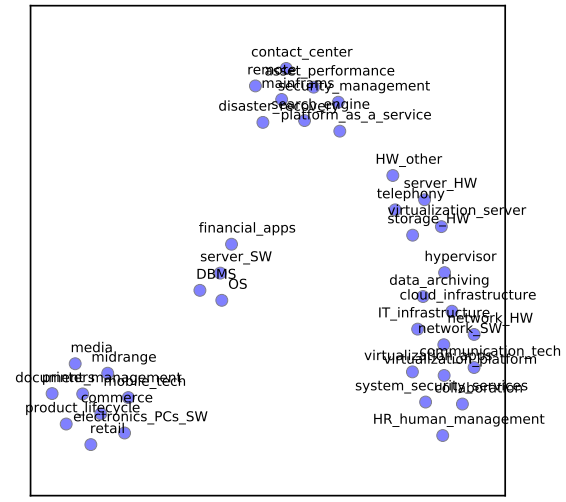


**Figure 9: LDA4 product embeddings.**

## 6 SALES APPLICATION

We have deployed LDA-based company representations in our recommendation tool. The company similarity search is based on LDA company representations with the HG Data Company dataset as input. Recommendations are built using our internal datasets. The tool searches for top-$k$ similar companies that are calculated using their LDA representations based on HG input. As LDA training is not done in a streaming fashion, it is done offline and can be retrained on demand or when the concept shift is taken place. In addition to the global similarity search, the tool also provides the user with filtering capabilities based on industry, location, number of employees and revenue.

This tool is currently used for an internal recommendation system.

## 7 CONCLUSION AND DISCUSSIONS

In this work, we assessed several data modeling techniques for product-company modeling, company similarity matching and recommendation. Companies have been considered to be similar based on the closeness of the structure of their IT install base. Assuming intrinsic hierarchies between products, companies and possibly latent install base structures, we have compared several techniques from the NLP domain capable of learning this kind of hidden hierarchical structures. These are unsupervised modeling techniques, namely, Latent Dirichlet Allocation and Recurrent Neural Networks. Having evaluated different model architectures, we demonstrated that LDA with 2, 3 and 4 latent topics fits our data best. We applied the company features learned by LDA to determine the top-$k$ similar companies, assessed the recommendation capabilities of the methods and deployed the best performers in a recommendation tool.

The results show that though there is clear sequential nature in the data, still the techniques that does not take time into account perform the best. The reason of this may be due to the higher number of parameters to learn in sequential techniques and the fact that our corpus is not enough to train all the parameters.

As future work, we will gather additional internal data about the IT structure of companies with proper timestamps and assess other deep neural network architectures starting from lower levels of product descriptions. We believe that, because of their hierarchical nature, deep neural networks should be able to discover hidden structures in IT install bases of companies if sufficient training data is provided.

## 8 ACKNOWLEDGMENTS

We would like to thank Viktor Kuropiatnyk for helping us make sense of D-U-N-S® data; Michel Speiser and Daniel Bauer for their algorithm of company name matching, which we used for record linkage. Katsiaryna Mirylenka also thanks Daniil Mirylenka for the fruitful discussions during the idea crystallization process.

## REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] Charu C. Aggarwal. 2015. *Data Mining - The Textbook.* Springer.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of Machine Learning Research* 3 (2003), 993–1022.

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014).

[5] Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. *ACL 2013* (2013), 100.

[6] Michele Dallachiesa, Besmira Nushi, Katsiaryna Mirylenka, and Themis Palpanas. 2011. Similarity matching for uncertain time series: Analytical and experimental comparison. In *Proceedings of the 2nd ACM SIGSPATIAL QUeST*. 8–15.

[7] Michele Dallachiesa, Besmira Nushi, Katsiaryna Mirylenka, and Themis Palpanas. 2012. Uncertain Time-Series Similarity: Return to the Basics. *PVLDB* 5, 11 (2012), 1662–1673.

[8] Dun & Brddstreet Inc 2017. Dun & Bradstreet Company. http://www.dnb.com. (2017).

[9] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* (2016).

[10] J. A. Hartigan. 1972. Direct Clustering of a Data Matrix. *J. Amer. Statist. Assoc.* 67, 337 (1972), 123 – 129.

[11] R. Heckel and M. Vlachos. 2016. Interpretable recommendations via overlapping co-clusters. *ArXiv e-prints* (April 2016).

[12] HG Data Company 2017. HG Data Company. http://www.hgdata.com. (2017).

[13] Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of ACM SIGIR*. ACM, 50–57.

[14] Tommi S. Jaakkola and David Haussler. 1999. Exploiting Generative Models in Discriminative Classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*. MIT Press, Cambridge, MA, USA, 487–493.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. 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 2013*. 3111–3119.

[17] Katsiaryna Mirylenka, Graham Cormode, Themis Palpanas, and Divesh Srivastava. 2015. Conditional heavy hitters: Detecting interesting correlations in data streams. *The VLDB Journal* 24, 3 (2015), 395–414.

[18] Katsiaryna Mirylenka, Christoph Miksovic, and Paolo Scotton. 2016a. Applicability of Latent Dirichlet Allocation for Company Modeling. In *Industrial Conference on Data Mining (ICDM'2016)*. 55–60.

[19] Katsiaryna Mirylenka, Christoph Miksovic, and Paolo Scotton. 2016b. Recurrent neural networks for modeling company-product time series. In *2nd ECML/PKDD Workshop AALTD*.

[20] K. Mirylenka, T. Palpanas, G. Cormode, and D. Srivastava. 2013. Finding interesting correlations with conditional heavy hitters. In *IEEE 29th International Conference on Data Engineering (ICDE'2013)*. 1069–1080.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Anand Rajaraman and Jeffrey David Ullman. 2011. Data Mining. In *Mining of Massive Datasets*. Cambridge University Press, 1–17.

[23] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.

[24] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

[25] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. ACM, 880–887.

[26] L. van der Maaten and G.E. Hinton. 2008. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[27] Michail Vlachos, Francesco Fusco, Charalambos Mavroforakis, Anastasios Kyrillidis, and Vassilios G Vassiliadis. 2014. Improving co-cluster quality with application to product recommendations. In *CIKM*. ACM, 679–688.

[28] Chyi-Kwei Yau. 2017. Recommend. https://github.com/chyikwei/recommend. (2017).

[29] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).