Learning Latent Vector Spaces for Product Search

Van Gysel, C.; de Rijke, M.; Kanoulas, E.

Published in:
CIKM'16

DOI:
10.1145/2983323.2983702

Learning Latent Vector Spaces for Product Search

Christophe Van Gysel
cvangysel@uva.nl

Maarten de Rijke
derijke@uva.nl

Evangelos Kanoulas
e.kanoulas@uva.nl

University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

We introduce a novel latent vector space model that jointly learns the latent representations of words, e-commerce products and a mapping between the two without the need for explicit annotations. The power of the model lies in its ability to directly model the discriminative relation between products and a particular word. We compare our method to existing latent vector space models (LSI, LDA and word2vec) and evaluate it as a feature in a learning to rank setting. Our latent vector space model achieves its enhanced performance as it learns better product representations. Furthermore, the mapping from words to products and the representations of words benefit directly from the errors propagated back from the product representations during parameter estimation. We provide an in-depth analysis of the performance of our model and analyze the structure of the learned representations.

Keywords

Entity retrieval; Latent space models; Representation learning

1. INTRODUCTION

Retail through online channels has become an integral part of consumers' lives [40]. In addition to using these online platforms that generate hundreds of billions of dollars in revenue [22], consumers increasingly participate in multichannel shopping where they research items online before purchasing them in brick-and-mortar stores. Search engines are essential for consumers to be able to make sense of these large collections of products available online [30]. In the case of directed searching (in contrast to exploratory browsing), users formulate queries using characteristics of the product they are interested in (e.g., terms that describe the product's category) [51]. However, it is widely known that there exists a mismatch between queries and product representations where both use different terms to describe the same concepts [34]. Thus, there is an urgent need for better semantic matching methods.

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10. . . $15.00

DOI: http://dx.doi.org/10.1145/2983323.2983702

Product search is a particular example of the more general entity finding task that is increasingly being studied. Other entity finding tasks considered recently include searching for people [6], books [23] and groups [35]. Products are retrievable entities where every product is associated with a description and one or more user reviews. Therefore, we use the terms "product" and "entity" interchangeably in this paper. However, there are two important differences between product search and the entity finding task as defined by de Vries et al. [14]. First, in entity finding one retrieves entities of a particular type from large broad coverage multi-domain knowledge bases such as Wikipedia [3, 14]. In contrast, product search engines operate within a single domain which can greatly vary in size. Second, user queries in product search consist of free-form text [51], as opposed to the semi-structured queries with additional type or relational constraints being used in entity finding [5, 14].

In this paper we tackle the problem of discriminating between products based on the language (i.e., descriptions and reviews) they are associated with. Existing methods that are aimed at discriminating between entities based on textual data learn word representations using a language modeling objective or heuristically construct entity representations [16, 57]. Our approach directly learns two things: a unidirectional mapping between words and entities, as well as distributed representations of both words and entities. It does so in an unsupervised and automatic manner such that words that are strongly evidential for particular products are projected nearby those products. While engineering of representations is important in information retrieval [2, 10, 12, 16, 25, 61], unsupervised joint representation learning of words and entities has not received much attention. We fill this gap. Our focus on learning representations for an end-to-end task such as *product search* is in contrast to the large volume of recent literature on word representation learning [56] that has a strong focus on upstream components such as distributional semantics [42, 49], parsing [13, 56] and information extraction [13, 56]. In addition, our focus on *unsupervised* representation learning is in contrast to recent entity representation learning methods [10, 61] that heavily depend on precomputed entity relationships and cannot be applied in their absence.

In recent years, significant progress has been made concerning semantic representations of entities. We point out three key insights on which we build: (1) Distributed representations [27] learned by discriminative neural networks reduce the curse of dimensionality and improve generalization. Latent features encapsulated by the model are shared by different concepts and, consequently, knowledge about one concept influences knowledge about others. (2) Discriminative approaches outperform generative models if enough training data is available [7, 47] as discriminative models solve the classification problem directly instead of solving a more general problem first [58]. (3) Recently proposed unsu-

pervised neural retrieval models [57] do not scale as they model a distribution over all retrievable entities; the approach is infeasible during training if the collection of retrievable entities is large.

Building on these insights, we introduce Latent Semantic Entities (LSE), a method that learns separate representations of words and retrievable objects jointly for the case where mostly unstructured documents are associated with the objects (i.e., descriptions and user reviews for products) and without relying on predefined relationships between objects (e.g., knowledge graphs). LSE learns to discriminate between entities for a given word sequence by mapping the sequence into the entity representation space. Contrary to heuristically constructed entity representations [16], LSE learns the relationship between words and entities directly using gradient descent. Unlike [57], we avoid computing the full probability distribution over entities; we do so by using noise-contrastive estimation.

Our research questions are as follows: (1) How do the parameters of LSE influence its efficacy? (2) How does LSE compare to latent vector models based on LDA, LSI and word2vec? (3) How does LSE compare to a smoothed language model that applies lexical term matching? (4) What is the benefit of incorporating LSE as a feature in a learning-to-rank setting?

We contribute: (1) A latent vector model, LSE, that jointly learns the representations of words, entities and the relationship between the former, together with an open-source implementation.[1] (2) A study of the influence of LSE's parameters and how these influence its ability to discriminate between entities. (3) An in-depth comparative analysis of the entity retrieval effectiveness of latent vector models. (4) Insights in how LSE can improve retrieval performance in entity-oriented search engines. (5) An analysis of the differences in performance between latent vector models by examining entity representations and mappings from queries to entity space.

2. RELATED WORK

2.1 Product retrieval

Product search engines are an important source of traffic in the e-commerce market [30]. Specialized solutions are needed to maximize the utilization of these platforms. Nurmi et al. [48] note a discrepancy between buyers' shopping lists and how retail stores maintain information. They introduce a grocery retrieval system that retrieves products using shopping lists written in natural language. Product resolution [1] is an important task for e-commerce aggregation platforms, such as verticals of major web search engines and price comparison websites. Duan et al. [19] propose a probabilistic mixture model for the attribute-level analysis of product search logs. They focus on structured aspects of product entities, while in this work we learn representations from unstructured documents. Duan et al. [20] extend the language modeling approach to product databases by incorporating the ability to condition on specification (e.g., lightweight products only). They note that while languages such as SQL can be used effectively to query these databases, their use is difficult for non-experienced end users. Duan and Zhai [18] study the problem of learning query intent representation for structured product entities. They emphasize that existing methods focus only on the query space and overlook critical information from the entity space and the connection in between.

We agree that modeling the connection between query words and entities and propagating information from the entity representations back to words is essential. In contrast to their work, we consider the problem of learning representations for entities based on their associations with unstructured documents.

[1] https://github.com/cvangysel/SERT

2.2 Latent semantic information retrieval

The mismatch between queries and documents is a critical challenge in search [34]. Latent Semantic Models (LSMs) enable retrieval based on conceptual content, instead of exact word matches. LSMs have become popular through the introduction of Latent Semantic Indexing (LSI) [15], followed by probabilistic LSI (pLSI) [28]. Salakhutdinov and Hinton [52] use a deep auto-encoder for the unsupervised learning of latent semantic document bit patterns. Deep Structured Semantic Models [29, 54] employ click data to predict a document's relevance to a query. Methods based on neural networks have also been used for machine-learned ranking [11, 17, 36]. Van Gysel et al. [57] introduce an LSM for entity retrieval, with an emphasis on expert finding; they remark that training the parameters of their model becomes infeasible when the number of entities increases. In this work we mitigate this problem by considering only a random sample of entities as negative examples during training. This allows us to efficiently estimate model parameters in large product retrieval collections, which is not possibly using the approach of [57] due to its requirement to compute a normalization constant over all entities.

2.3 Representation learning

Recently, there has been a growing interest in neural probabilistic language models (LMs) for the modeling of word sequences [8, 43, 44]. Distributed representations [27] of words learned by neural LMs, also known as word embeddings, incorporate syntactic and semantic information [42, 45, 49] as a side-effect of their ability to reduce the dimensionality. Feed-forward [13] and recurrent [42] neural networks perform well in various NLP tasks. Very recently, there has been an increased interest in multimodal neural language models [33], which are used for the task of automated image captioning, amongst others. Learning representations of entities is not new. Bordes et al. [10] leverage structured relations captured in Knowledge Bases (KB) for entity representation learning and evaluate their representations on the link prediction task. Our approach has a strong focus on modeling the language of all entities collaboratively, without the need for explicit entity relations during training. Zhao et al. [61] employ matrix factorization methods to construct low-dimensional continuous representations of entities, categories and words for determining similarity of Wikipedia entities. They employ a word pair similarity evaluation set and only evaluate on pairs referring to Wikipedia entities; they learn a single semantic space for widely-differing concepts (entities, categories and words) of different cardinalities and make extensive use of an underlying Knowledge Graph (KG) to initialize their parameters. In contrast, we model representations of words and entities jointly in separate spaces, in addition to a mapping from word to entity representations, in an unsupervised manner.

We tackle the task of learning latent continuous vector representations for e-commerce products for the purpose of product search. The focus of this work lies in the language modeling and representation learning challenge. We learn distributed representations [27] of words and entities and a mapping between the two. At retrieval time, we rank entities according to the similarity of their latent representations to the projected representation of a query. Our model LSE is compared against existing entity-oriented latent vector representations that have been created using LSI, LDA and word2vec. We provide an analysis of model parameters and give insight in the quality of the joint representation space.

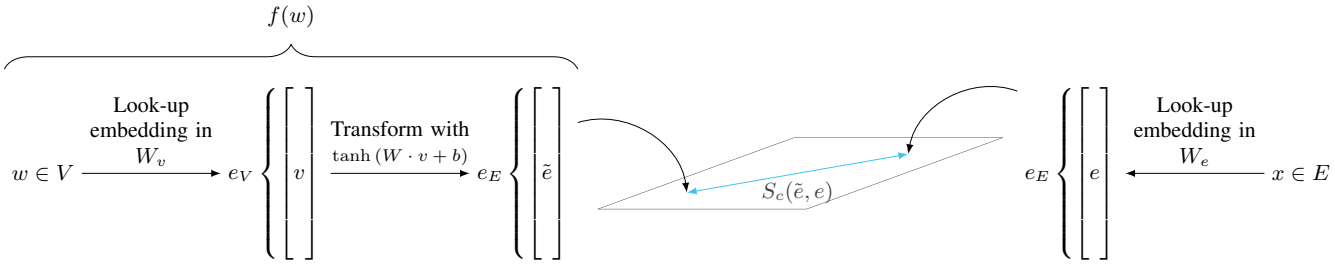


Figure 2: Schematic representation of the Latent Semantic Entities model for a single word w. Word embeddings W_v (e_V-dim. for $|V|$ words), entity embeddings W_e (e_E-dim. for $|X|$ entities) and the mapping from words to entities (e_E-by-e_V matrix W, e_E-dim. vector b) are learned using gradient descent.

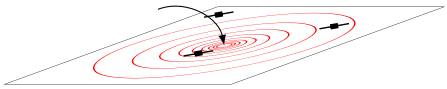


Figure 1: Illustrative example of how entities are ranked in vector space models w.r.t. a projected query. Query q is projected into entity space E using mapping f (black arrow) and entities (black crosses) are ranked according to their similarity in decreasing order.

3. LATENT VECTOR SPACES FOR ENTITY RETRIEVAL

We first introduce a generalized formalism and notation for entity-oriented latent vector space models. After that, in §3.2, we introduce Latent Semantic Entities, a latent vector space model that jointly learns representations of words, entities and a mapping between the two directly, based on the idea that entities are characterized by the words they are associated with and vice versa. Product representations are constructed based on the n-grams the products are likely to generate based on their description and reviews, while word representations are based on the entities they are associated with and the context they appear in. We model the relation between word and product representations explicitly so that we can predict the product representation for a previously unseen word sequence.

3.1 Background

We focus on a product retrieval setting in which a user wants to retrieve the most relevant products on an e-commerce platform. As in typical information retrieval scenarios, the user encodes their information need as a query q and submits it to a search engine. Product search queries describe characteristics of the product the user is searching for, such as a set of terms that describe the product's category [51].

Below, X denotes the set of entities that we consider. For every $x_i \in X$ we assume to have a set of associated documents D_{x_i}. The exact relation between the entity and its documents depends on the problem setting. In this paper, entities are products [18, 48] and documents associated with these products are descriptions and product reviews.

Latent vector space models rely on a function $f : V^+ \to E$ that maps a sequence of words (e.g., a query q during retrieval) from a vocabulary V to a e_E-dimensional continuous entity vector space $E \subset \mathbb{R}^{e_E}$. Every entity $x_i \in X$ has a corresponding vector representation $e_i \in E$. Let $S_c : E \times E \to \mathbb{R}$ denote the cosine similarity between vectors in E. For a given query q, entities x_i are ranked in decreasing order of the cosine similarity between e_i and the query projected into the space of entities, $f(q)$. Fig. 1 illustrates how entities are ranked according to a projected query. For LSI, f is defined as the multiplication of the term-frequency vector representation of q with the rank-reduced term-concept matrix and the inverse of the rank-reduced singular value matrix [15]. In the case of LDA, f becomes the distribution over topics conditioned on q [9]. This distribution is computed as the sum of the topic distributions conditioned on the individual words of q. In this paper, the embedding f is learned; see §3.3 below.

Traditional vector space models operate on documents instead of entities. Demartini et al. [16] extend document-oriented vector spaces to entities by representing an entity as a weighted sum of the representations of their associated documents:

$$e_i = \sum_{d_j \in D_{x_i}} r_{i,j} f(d_j) \tag{1}$$

where $f(d_j)$ is the vector representation of d_j and $r_{i,j}$ denotes the relationship weight between document d_j and entity x_i. In this work we put $r_{i,j} = 1$ whenever $d_j \in D_{x_i}$ for a particular $x_i \in X$ and $r_{i,j} = 0$ otherwise, as determining the relationship weight between entities and documents is a task in itself.

3.2 Latent semantic entities

While Eq. 1 adapts document-oriented vector space models to entities, in this work we define f by explicitly learning (§3.3) the mapping between word and entity representations and the representations themselves:

$$f(s) = \tanh\left(W \cdot (W_v \cdot \frac{1}{|s|} \sum_{w_i \in s} \delta_i) + b\right) \tag{2}$$

for a string s of constituent words $w_1, \ldots, w_{|s|}$ (an n-gram extracted from a document or a user-issued query), where W_v is the $e_V \times |V|$ projection matrix that maps the averaged one-hot representations (i.e., a $|V|$-dimensional vector with element i turned on and zero elsewhere) of word w_i, δ_i, to its e_V-dimensional distributed representation. This is equivalent to taking the embeddings of the words in s and averaging them. In addition, b is a e_E-dimensional bias vector, W is the $e_E \times e_V$ matrix that maps averaged word embeddings to their corresponding position in entity space E and $\tanh$ is the element-wise smooth hyperbolic tangent with range $(-1, 1)$. This transformation allows word embeddings and entity embeddings to be of a different dimensionality.

In other words, for a given string of words we take the representation of this string to be the average of the representations of the words it contains [42, 50]. This averaged word representation is then transformed using a linear map (W) and afterwards translated using b. We then apply the hyperbolic tangent as non-linearity such that every component lies between -1 and 1. First of all, this regularizes the domain of the space and avoids numerical instability issues that occur when the magnitude of the vector components be-

comes too large. Secondly, by making the function non-linear we are able to model non-linear class boundaries in the optimization objective that we introduce in the next section. We use W_e to denote the $|X| \times e_E$ matrix that holds the entity representations. Row i of W_e corresponds to the vector representation, e_i, of entity x_i. Fig. 2 depicts a schematic overview of the proposed model. The parameters W_v, W, b and W_e will be learned automatically using function approximation methods as explained below.

The model proposed in this section shares similarities with previous work on word embeddings and unsupervised neural retrieval models [42, 57]. However, its novelty lies in its ability to scale to large collections of entities and its underlying assumption that words and entities are embedded in spaces of different dimensionality: (1) The model of [42] has no notion of entity retrieval as it estimates a language model for the whole corpus. (2) Similar to [42], Eq. 2 aggregates words $w_i \in s$ to create a single phrase representation of s. However, in [57], a distribution $P(X \mid w_i)$ is computed for every w_i independently and aggregation occurs using the factor product. This is infeasible during model training when the collection of retrievable objects becomes too large, as is the case for product search. In the next section (§3.3) we solve this problem by sampling. (3) In both [42, 57] two sets of representations of the same dimensionality are learned for different types of objects with potentially different latent structures (e.g., words, word contexts and experts). As mentioned earlier, Eq. 2 alleviates this problem by transforming one latent space to the other.

3.3 Parameter estimation

For a particular document $d \in D_{x_i}$ associated with entity x_i, we generate n-grams $w_{j,1}, \ldots, w_{j,n}$ where n (window size) remains fixed during training. For every n-gram $w_{j,1}, \ldots, w_{j,n}$, we compute its projected representation $f(w_{j,1}, \ldots, w_{j,n})$ in E using f (Eq. 2). The objective, then, is to directly maximize the similarity between the vector representation of the entity e_i and the projected n-gram $f(w_{j,1}, \ldots, w_{j,n})$ with respect to S_c (§3.1), while minimizing the similarity between $f(w_{j,1}, \ldots, w_{j,n})$ and the representations of non-associated entities. This allows the model to learn relations between neighboring words in addition to the associated entity and every word.

However, considering the full set of entities for the purpose of discriminative training can be costly when the number of entities $|X|$ is large. Therefore, we apply a variant of Noise-Contrastive Estimation (NCE) [26, 41, 45, 46] where we sample negative instances from a noise distribution with replacement. We use the uniform distribution over entities as noise distribution. Define

$$P(\mathcal{S} \mid e_i, f(w_{j,1}, \ldots, w_{j,n})) = \sigma(e_i \cdot f(w_{j,1}, \ldots, w_{j,n})) \quad (3)$$

as the similarity of two representations in latent entity space, where

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

denotes the sigmoid function and $\mathcal{S}$ is an indicator binary random variable that says whether x_i is similar to $f(w_{j,1}, \ldots, w_{j,n})$.

We then approximate the probability of an entity x_i given an n-gram by randomly sampling z contrastive examples:

$$\begin{aligned} &\log \tilde{P}(x_i \mid w_{j,1}, \ldots, w_{j,n}) \quad (4) \\ &= \log P(\mathcal{S} \mid e_i, f(w_{j,1}, \ldots, w_{j,n})) \\ &\quad + \sum_{\substack{k=1, \\ x_k \sim U(X)}}^{z} \log\left(1 - P(\mathcal{S} \mid e_k, f(w_{j,1}, \ldots, w_{j,n}))\right) \end{aligned}$$

where $U(X)$ denotes the uniform distribution over entities X, the noise distribution used in NCE [26]. Eq. 4 avoids iterating over all entities during parameter estimation as we stochastically sample z entities uniformly as negative training examples.[2]

During model construction we maximize the log-probability (4) using batched gradient descent. The loss function for a single batch of m instances $((w_{k,1}, \ldots, w_{k,n}), x_k)$ consisting of n-grams sampled from documents D_{x_k} (see §4.2) and associated entity x_k is as follows:

$$\begin{aligned} &L(W_v, W_e, W, b) \\ &= -\frac{1}{m} \sum_{k=1}^{m} \log \tilde{P}(x_k \mid w_{k,1}, \ldots, w_{k,n}) \\ &\quad + \frac{\lambda}{2m} \left(\sum_{i,j} {W_v}^2_{i,j} + \sum_{i,j} {W_e}^2_{i,j} + \sum_{i,j} W^2_{i,j} \right), \quad (5) \end{aligned}$$

where λ is a weight regularization parameter. Instances are shuffled before batches are created. The update rule for a particular parameter θ (W_v, W_e, W or b) given a single batch of size m is:

$$\theta^{(t+1)} = \theta^{(t)} - \boldsymbol{\alpha}^{(t)} \odot \frac{\partial L}{\partial \theta}({W_v}^{(t)}, {W_e}^{(t)}, W^{(t)}, b^{(t)}), \quad (6)$$

where $\boldsymbol{\alpha}^{(t)}$ and $\theta^{(t)}$ denote the per-parameter learning rate and parameter θ at time t, respectively. The learning rate $\boldsymbol{\alpha}$ consists of the same number of elements as there are parameters; in the case of a global learning rate, all elements of $\boldsymbol{\alpha}$ are equal to each other. The derivatives of the loss function (5) are given in the Appendix.

4. EXPERIMENTAL SETUP

4.1 Research questions

In this paper we investigate the problem of constructing a latent vector model of words and entities by directly modeling the discriminative relation between entities and word context. We seek to answer the following research questions:

RQ1 How do the parameters of LSE influence its efficacy?

In §3 we introduced various hyper-parameters along with the definition of Latent Semantic Entities. We have the size of word representations e_V and the dimensionality of the entity representations e_E. During parameter estimation, the window size n influences the context width presented as evidence for a particular entity. What is the influence of these parameters on the effectiveness of LSE and can we identify relations among parameters?

RQ2 How does LSE compare to latent vector models based on LDA, LSI and word2vec?

Is there a single method that always performs best or does effectiveness differ per domain? Does an increase in the vector space dimensionality impact the effectiveness of these methods?

RQ3 How does LSE compare to a smoothed language model that applies lexical term matching?

How does LSE compare to language models on a per-topic basis? Are there particular topics that work especially well with either type of ranker?

RQ4 What is the benefit of incorporating LSE as a feature in a learning-to-rank setting?

[2]We exploit the special nature of our evaluation scenario where we know the unique association between documents and entities. The setup can easily be adapted to the more general case where a document is associated with multiple entities by extracting the same word sequences from the document for every associated entity.

What if we combine popularity-based, exact matching and latent vector space features in a linear learning-to-rank setting? Do we observe an increase in effectiveness if we combine these features?

4.2 Experimental design

To answer the research questions posed in §4.1, we evaluate LSE in an entity retrieval setting organized around Amazon products (see §4.3). We choose to experiment with samples of Amazon product data [38, 39] for the following reasons: (1) The collection contains heterogeneous types of evidential documents associated with every entity: descriptions as well as reviews. (2) Every department (e.g., *Home & Kitchen*) constitutes a separate, self-contained domain. (3) Within each department there is a hierarchical taxonomy that partitions the space of entities in a rich structure. We can use the labels associated with these partitions and the partitions themselves as ground truth during evaluation. (4) Every department consists of a large number of products categorized over a large number of categories. Importantly, this allows us to construct benchmarks with an increasing number of entities. (5) Every product has a variety of attributes that can be used as popularity-based features in a learning-to-rank setting.

To answer **RQ1** we investigate the relation between the dimensionality of the entity representations e_E and window size n. The latter, the window size n, controls the context width the model can learn from, while the former, the dimensionality of the entity representations e_E, influences the number of parameters and expressive power of the model. We sweep exponentially over n (2^i for $0 \leq i < 6$) and e_E (2^i for $6 \leq i < 11$). **RQ2** is answered by comparing LSE with latent vector space model baselines (§4.5) for an increasing entity space dimensionality e_E (2^i for $6 \leq i < 11$). For **RQ3**, we compare the per-topic paired differences between LSE and a lexical language model. In addition, we investigate the correlation between lexical matches in relevant entity documents and ranker preference. We address **RQ4** by evaluating LSE as a feature in a machine-learned ranking in addition to query-independent and lexical features.

The number of n-grams sampled per entity $x \in X$ from associated documents D_x in every epoch (i.e., iteration of the training data) is equal to $\left\lceil \frac{1}{|X|} \sum_{d \in D} \max\left(|d| - n + 1, 0\right) \right\rceil$,where the $|\cdot|$ operator is used interchangeably for the size of set X and the number of tokens in documents $d \in D$. This implicitly imposes a uniform prior over entities (i.e., stratified sampling where every entity is of equal importance). The word vocabulary V is created for each benchmark by ignoring punctuation, stop words and case; numbers are replaced by a numerical placeholder token. We prune V by only retaining the 2^{16} most-frequent words so that each word can be encoded by a 16-bit unsigned integer. In terms of parameter initialization of the Latent Semantic Entities model, we sample the initial matrices W_v, W (Eq. 2) and W_e uniformly in the range $\left[-\sqrt{\frac{6.0}{m+n}}, \sqrt{\frac{6.0}{m+n}}\right]$ for an $m \times n$ matrix, as this initialization scheme is known to improve model training convergence [24], and take the bias vector b to be null. The number of word features is set to $e_V = 300$, similar to [41]. We take the number of negative examples $z = 10$ to be fixed. Mikolov et al. [41] note that a value of z between 10 and 20 is sufficient for large data sets [45].

We used Adam ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) [32] with batched gradient descent ($m = 4096$) and weight decay $\lambda = 0.01$ during training on NVidia Titan X GPUs. Adam has been designed specifically for non-stationary, stochastic cost functions like the one we defined in Eq. 4. For every model, we iterate over the training data 15 times and choose the best epoch based on the validation sets (Table 1).

4.3 Product search benchmarks

We evaluate on four samples from different product domains[3] (Amazon departments), each with of an increasing number of products: *Home & Kitchen* (8,192 products), *Clothing, Shoes & Jewelry* (16,384 products), *Pet Supplies* (32,768 products) and *Sports & Outdoors* (65,536 products); see Table 1. The documents associated with every product consist of the product description plus reviews provided by Amazon customers.

Rowley [51, p. 24] describes directed product search as users searching for "a producer's name, a brand or a set of terms which describe the category of the product." Following this observation, the test topics c_i are extracted from the categories each product belongs to. Category hierarchies of less than two levels are ignored, as the first level in the category hierarchy is often non-descriptive for the product (e.g., in *Clothing, Shoes & Jewelry* this is the gender for which the clothes are designated). Products belonging to a particular category hierarchy are considered as relevant for its extracted topic. Products can be relevant for multiple topics. Textual representations q_{c_i} of the topics based on the categories are extracted as follows. For a single hierarchy of categories, we tokenize the titles of its sub-categories and remove stopwords and duplicate words. For example, a digital camera lense found in the *Electronics* department under the categorical topic *Camera & Photo* → *Digital Camera Lenses* will be relevant for the textual query "*photo camera lenses digital*." Thus, we only have two levels of relevance. We do not index the categories of the products as otherwise the query would match the category and retrieval would be trivial.

4.4 Evaluation measures and significance

To measure retrieval effectiveness, we report Normalized Discounted Cumulative Gain (NDCG). For **RQ4**, we additionally report Precision@k ($k = 5, 10$). Unless mentioned otherwise, significance of observed differences is determined using a two-tailed paired Student's t-test [55] ($^{***}\, p < 0.01$; $^{**}\, p < 0.05$; $^{*}\, p < 0.1$).

4.5 Methods used in comparisons

We compare Latent Semantic Entities to state-of-the-art latent vector space models for entity retrieval that are known to perform semantic matching [34]. We also conduct a contrastive analysis between LSE and smoothed language models with exact matching capabilities.

Vector Space Models for entity finding. Demartini et al. [16] propose a formal model for finding entities using document vector space models (§3.1). We compare the retrieval effectiveness of LSE with baseline latent vector space models created using (1) Latent Semantic Indexing (LSI) [15] with TF-IDF term weighting, (2) Latent Dirichlet Allocation (LDA) [9] with $\alpha = \beta = 0.1$, where a document is represented by its topic distribution, and (3) word2vec [42] with CBOW and negative sampling, where a query/document is represented by the average of its word embeddings (same for queries in LSE). Similar to LSE, we train word2vec for 15 iterations and select the best-performing model using the validation sets (Table 1).

Query-likelihood Language Model. For every entity a profile-based statistical language model is constructed using maximum-likelihood estimation [4, 37, 58], which is then smoothed by the language model of the entire corpus. The retrieval score of entity x for query q is defined as

$$\tilde{P}(q \mid x) = \prod_{t_i \in q} P(t_i \mid \theta_x), \tag{7}$$

[3] A list of product identifiers, topics and relevance assessments can be found at https://github.com/cvangysel/SERT.

Table 1: Overview of the *Home & Kitchen*, *Clothing, Shoes & Jewelry*, *Pet Supplies* and *Sports & Outdoors* product search benchmarks. T and V denote the test and validation sets, respectively. Arithmetic mean and standard deviation are reported wherever applicable.

	Home & Kitchen	Clothing, Shoes & Jewelry	Pet Supplies	Sports & Outdoors
Corpus (train)				
Number of documents	88,130	94,024	416,993	502,313
Document length	70.02 ± 73.82	58.41 ± 61.90	77.48 ± 78.44	72.52 ± 81.47
Number of entities	8,192	16,384	32,768	65,536
Documents per entity	10.76 ± 52.01	5.74 ± 18.60	12.73 ± 55.98	7.66 ± 30.38
Topics (test)				
Topics	657 (T) 72 (V)	750 (T) 83 (V)	385 (T) 42 (V)	1,879 (T) 208 (V)
Terms per topic	5.11 ± 1.79	4.10 ± 1.86	3.73 ± 1.62	4.64 ± 1.68
Relevant entities per topic	10.92 ± 32.41 (T) 10.29 ± 15.66 (V)	20.15 ± 57.78 (T) 12.13 ± 19.85 (V)	75.96 ± 194.44 (T) 57.40 ± 88.91 (V)	29.27 ± 61.71 (T) 38.25 ± 157.34 (V)

where $P(t \mid \theta_x)$ is the probability of term t occurring in the smoothed language model of x (Jelinek-Mercer smoothing [60]). Given a query q, entities are ranked according to $\tilde{P}(q \mid x)$ in descending order.

Machine-learned ranking. RankSVM models [31] in §5.2 and 6 are trained using stochastic gradient descent using the implementation of Sculley [53]. We use default values for all parameters, unless stated otherwise. For the experiment investigating LSE as a feature in machine-learned ranking in §5.2, we construct training examples by using the relevant entities as positive examples. Negative instances are generated by sampling from the non-relevant entities with replacement until the class distribution is uniform.

5. RESULTS AND DISCUSSION

We start by giving a high-level overview of our experimental results (**RQ1** and **RQ2**), followed by a comparison with lexical matching methods (**RQ3**) and the use of LSE as a ranking feature (**RQ4**) (see §4.2 for an overview of the experimental design).

5.1 Overview of experimental results

RQ1: Fig. 3 depicts a heat map for every combination of window size and entity space dimensionality evaluated on the validation sets (Table 1). Fig. 3 shows that neither extreme values for the dimensionality of the entity representations nor the context width alone achieve the highest performance on the validation sets.

Instead, a low-dimensional entity space (128- and 256-dimensional) combined with a medium-sized context window (4- and 8-grams) achieve the highest NDCG. In the two largest benchmarks (Fig. 3c, 3d) we see that for 16-grams, NDCG actually lowers as the dimensionality of the entity space increases. This is due to the model *fitting* the optimization objective (Eq. 5), which we use as an unsupervised surrogate of relevance, too well. That is, as the model is given more learning capacity (i.e., higher dimensional representations), it starts to learn more regularities of natural language which counteract retrieval performance.

RQ2: Fig. 4 presents a comparison between LSE (window size $n = 4$) and vector space model baselines (§4.5) for increasing entity representation dimensionality (2^i for $6 \leq i < 11$) on the test sets. LSE significantly outperforms ($p < 0.01$) all baseline methods in most cases (except for Fig. 4a where $e_E = 1024$). For the smaller benchmarks (Fig. 4a, 4b), we see LSI as the main competitor of LSE. However, as the training corpora become larger (in Fig. 4c, 4d), word2vec outperforms LSI and becomes the main contester of LSE. On all benchmarks, LSE peaks when the entity representations are low-dimensional (128- or 256-dimensional) and afterwards (for a higher dimensionality) performance decreases. On the other hand, word2vec stagnates in terms of NDCG around representations of 512 dimensions and never achieves the same level as LSE did for one or two orders of magnitude (base 2) smaller representations. This is a beneficial trait of LSE, as high-dimensional vector spaces are undesirable due to their high computational cost during retrieval [59].

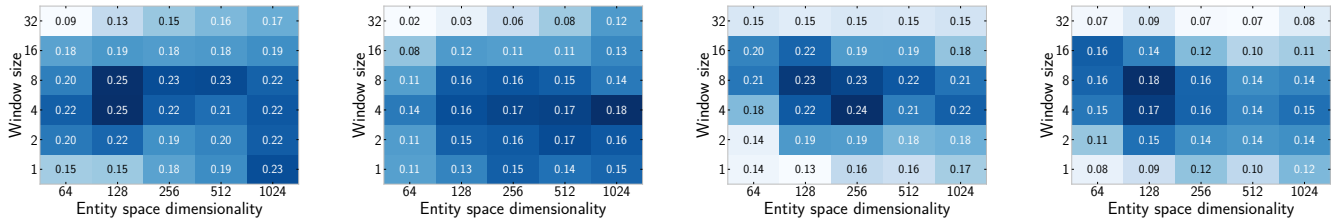
Table 2: Correlation coefficients between average IDF of lexically matched terms in documents associated with relevant entities and $\triangle$NDCG. A negative correlation coefficient implies that queries consisting of more specific terms (i.e., low document freq.) that occur exactly in documents associated with relevant entities are more likely to benefit from QLM, whereas other queries (with less specific terms or less exact matches) gain more from LSE. Significance is achieved for all benchmarks ($p < 0.01$) using a permutation test.

Benchmark	Spearman R	Pearson R
Home & Kitchen	-0.30	-0.35
Clothing, Shoes & Jewelry	-0.40	-0.37
Pet Supplies	-0.17	-0.17
Sports & Outdoors	-0.34	-0.36

5.2 A feature for machine-learned ranking

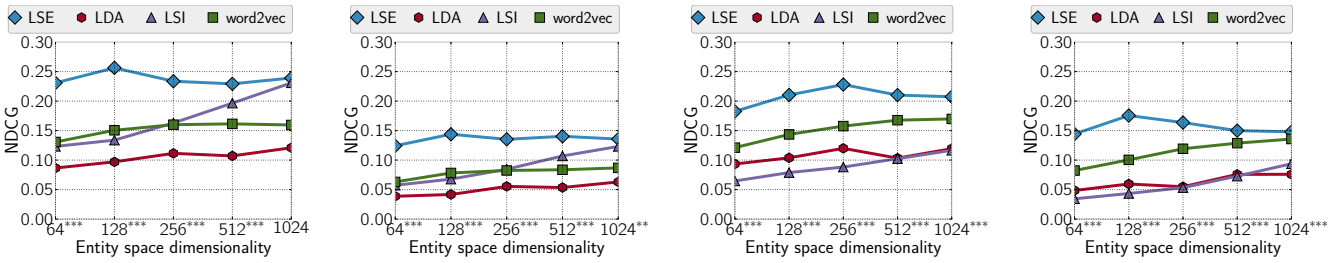
We now investigate the use of LSE as a feature in a learning to rank setting [36]. Latent vector space models are known to provide a means of semantic matching as opposed to a purely lexical matching [34, 57]. To determine to which degree this is indeed the case, we first perform a topic-wise comparison between LSE and a lexical language model, the Query-likelihood Language Model (QLM) [60], as described in §4.5. We optimize the parameters of LSE and QLM on the validation sets for every benchmark (Table 1). In the case of LSE, we select the model that performs best in Fig. 3. For QLM, we sweep over λ linearly from 0.0 to 1.0 (inclusive) with increments of 0.05.

RQ3: Fig. 5 shows the per-topic paired difference between LSE and QLM in terms of NDCG. Topics that benefit more from LSE have a positive value on the y-axis, while those that prefer QLM have a negative value. We can see that both methods perform similarly for many topics (where $\triangle = 0.0$). For certain topics one



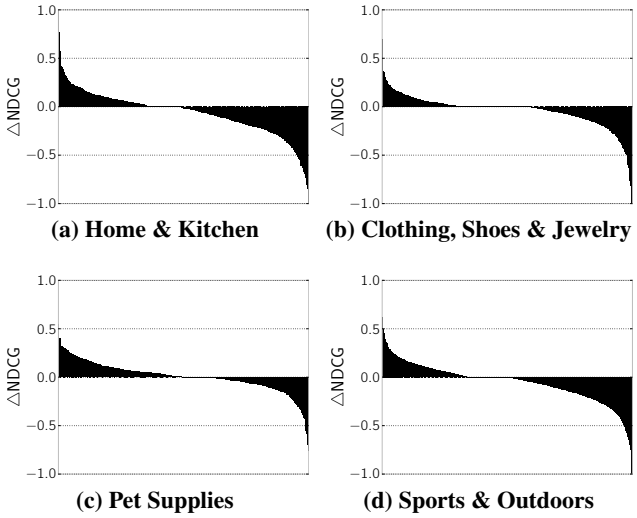
(a) Home & Kitchen **(b) Clothing, Shoes & Jewelry** **(c) Pet Supplies** **(d) Sports & Outdoors**

Figure 3: Sensitivity analysis of LSE in terms of NDCG for window size n and the size of entity representations e_E during parameter estimation (Eq. 5) for models trained on *Home & Kitchen*, *Clothing, Shoes & Jewelry*, *Pet Supplies* and *Sports & Outdoors* product search benchmarks (§4.3) and evaluated on the validation sets.



(a) Home & Kitchen **(b) Clothing, Shoes & Jewelry** **(c) Pet Supplies** **(d) Sports & Outdoors**

Figure 4: Comparison of LSE (with window size $n = 4$) with latent vector space baselines (LSI, LDA and word2vec; §4.5) on *Home & Kitchen*, *Clothing, Shoes & Jewelry*, *Pet Supplies* and *Sports & Outdoors* product search benchmarks (§4.3) and evaluated on the test sets. Significance (§4.4) is computed between LSE and the baselines for each vector space size.



(a) Home & Kitchen **(b) Clothing, Shoes & Jewelry**

(c) Pet Supplies **(d) Sports & Outdoors**

Figure 5: Per-topic paired differences between LSE and Query-likelihood Language Model for models trained on *Home & Kitchen*, *Clothing, Shoes & Jewelry*, *Pet Supplies* and *Sports & Outdoors* product search benchmarks (§4.3) and evaluated on the test sets. For every plot, the y-axis indicates △NDCG between LSE and a Query-likelihood Language Model. The x-axis lists the topics in the referenced benchmark in decreasing order of △NDCG such that topics for which LSE performs better are on the left and vice versa for the Query-likelihood Language Model on the right.

method performs substantially better than the other, suggesting that the two are complementary. To further quantify this, we investigate the relation between specific topic terms and their occurrence in documents relevant to these topics. That is, we measure the correlation between the per-topic △NDCG (as described above) and the average inverse document frequency (IDF) of exact/lexically matched terms in the profile-based language model. In Table 2 we observe that queries that contain specific tokens (i.e., with high inverse document frequency) and occur exactly in documents associated with relevant products, benefit more from QLM (lexical matches). Conversely, queries with less specific terms or without exact matches in the profiles of relevant products gain more from LSE (semantic matches).

This observation motivates the use of LSE as a ranking feature in addition to traditional language models. Specifically, we now evaluate the use of LSE as a feature in a linear RankSVM (§4.5). Following Fang et al. [21], we consider query-independent (QI) popularity-based features in addition to features provided by LSE and QLM. This allows us to consider the effect of the query-dependent features independent from their ability to model a popularity prior over entities. Table 3 lists the feature sets.

RQ4: Table 4 shows the results for different combinations of feature sets used in a machine-learned ranker, RankSVM. The experiment was performed using 10-fold cross validation on the test sets (Table 1). The combination using all features outperforms smaller subsets of features, on all metrics. We conclude that Latent Semantic Entities adds a signal that is complementary to traditional (lexical) language models, which makes it applicable in a wide range of entity-oriented search engines that use ranker fusion techniques.

Table 3: Overview of the feature sets used in the machine-learned ranking experiments.

Features	Description
QI	Query-independent features: (1) product price; (2) product description length; (3) reciprocal of the Amazon sales rank; and (4) product PageRank scores based on four related product graphs (also bought, also viewed, bought together, buy after viewing).
QLM	Query-likelihood Language Model using Jelinek-Mercer smoothing with λ optimized on the validation set (Table 1). Posterior $P(q \mid x)$ is used as a feature for entity x and query q.
LSE	Latent Semantic Entities optimized on the validation set (Table 1, Fig. 3). Similarity $S_c(f(q), e)$ is used as a feature for entity x, with vector representation e, and query q.

6. ANALYSIS OF REPRESENTATIONS

Next, we analyze the entity representations e_i of the vector space models independent of the textual representations by providing empirical lower-bounds on their maximal retrieval performance, followed by a comparison with their actual performance so as to measure the effectiveness of word-to-entity mapping f.

Fig. 3 and 4 show which levels of performance may be achieved by using the latent models to generate a ranking from textual queries (Eq. 2). But this is only one perspective. As entities are ranked according to their similarity with the projected query vector $f(q_c)$, the performance for retrieving entities w.r.t. the textual representation of a topic c depends on the structure of the entity space E, the ideal retrieval vector $e_c^* \in E$ (i.e., the vector that optimizes retrieval performance), and the similarity between $f(q_c)$ and e_c^*.

How can we determine the ideal vector e_c^*? First, we define it to be the vector for which the cosine similarity with each of the entity embeddings results in a ranking where relevant entities are ranked higher than non-relevant or unjudged entities. We approximate e_c^* by optimizing the pair-wise SVM objective [31, 53]. That is, for every topic c we construct a separate RankSVM model based on its ground-truth as follows. We only consider topics with at least two relevant entities, as topics with a single relevant entity have a trivial optimal retrieval vector (the entity representation of the single relevant entity). Using the notation of [31], the normalized entity representations are used as features, and hence the feature mapping ϕ is defined as

$$\phi(c, x_i) = \frac{e_i}{\|e_i\|_2} \text{ for all } x_i \in X.$$

The target ranking r_c^* is given by the entities relevant to topic c. Thus, the features for every entity become the entity's normalized representation and its label is positive if it is relevant for the topic and negative otherwise. The pair-wise objective then finds a weight vector such that the ranking generated by ordering according to the vector scalar product between the weight vector and the normalized entity representations correlates with the target ranking r_c^*. Thus, our approximation of the *ideal vector*, $\tilde{e}_c^*$, is given by the weight vector w_c for every c.[4]

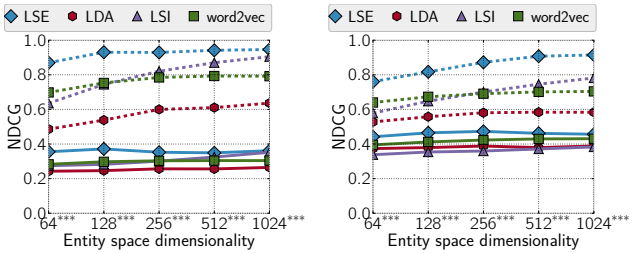
What is the performance of this approximately ideal vector representation? And how far are our representations removed from it? Fig. 6 shows the absolute performance of $\tilde{e}_c^*$ (dashed curves) and $f(q)$ (solid curves) in terms of NDCG. Comparing the (absolute) difference between every pair of dashed and solid curves for a single latent model gives an intuition of how much performance in terms of NDCG there is to gain by improving the projection function f for that method. The approximately ideal vectors $\tilde{e}_c^*$ discovered for LSE outperform all baselines significantly. Interestingly, for representations created using LDA, the optimal performance goes up while the actual performance stagnates. This indicates that a higher vector space dimensionality renders better representations using LDA, however, the projection function f is unable to keep up in the sense that projected query vectors are not similar to the representations of their relevant entities. The latent models with the best representations (LSE and LSI) also have the biggest gap between $f(q)$ and $\tilde{e}_c^*$ in terms of achieved NDCG.

We interpret the outcomes of our analysis as follows. The entity space E has more degrees of freedom to cluster entities more appropriately as the dimensionality of E increases. Consequently, the query projection function f is expected to learn a more complex function. In addition, as the dimensionality of E increases, so does the modeling capacity of the projection function f in the case of LSE and LSI (i.e., the transformation matrices become larger) and therefore more parameters have to be learned. We conclude that our method can more effectively represent entities in a lower-dimensional space than LSI by making better use of the vector space capacity. This is highly desirable, as the asymptotic runtime complexity of many algorithms operating on vector spaces increases at least linearly [59] with the size of the vectors.

Table 4: Ranking performance results for query independent (QI) features, the Query-likelihood Language Model (QLM) match feature, the Latent Semantic Entities (LSE) match feature and combinations thereof, weighted using RankSVM (§5.2), evaluated on the test sets using 10-fold cross validation, for *Home & Kitchen*, *Clothing, Shoes & Jewelry*, *Pet Supplies* and *Sports & Outdoors* product search benchmarks (§4.3). The hyperparameters of the individual query features (QLM and LSE) were optimized using the validation sets. Significance of the results (§4.4) is computed between QI + QLM + LSE and QI + QLM.

	Home & Kitchen		
	NDCG	P@5	P@10
QI	0.005	0.002	0.001
QI + QLM	0.321	0.180	0.145
QI + LSE	0.257	0.121	0.107
QI + QLM + LSE	**0.352*****	**0.192****	**0.157*****
	Clothing, Shoes & Jewelry		
	NDCG	P@5	P@10
QI	0.002	0.001	0.001
QI + QLM	0.177	0.079	0.068
QI + LSE	0.144	0.065	0.057
QI + QLM + LSE	**0.198*****	**0.094*****	**0.080*****
	Pet Supplies		
	NDCG	P@5	P@10
QI	0.003	0.002	0.002
QI + QLM	0.250	0.212	0.199
QI + LSE	0.268	0.222	0.214
QI + QLM + LSE	**0.298*****	**0.255*****	**0.236*****
	Sports & Outdoors		
	NDCG	P@5	P@10
QI	0.001	0.001	0.001
QI + QLM	0.235	0.183	0.156
QI + LSE	0.188	0.132	0.121
QI + QLM + LSE	**0.264*****	**0.192*****	**0.172*****

[4]Note that $\tilde{e}_c^*$ does not take into account the textual representations q_c of topic c, but only the clustering of entities relevant to c and their relation to other entities.



(a) Home & Kitchen (b) Pet Supplies

Figure 6: Comparison of the approximately ideal retrieval vector $\tilde{e}_c^*$ with the projected query retrieval vector $f(q)$ for latent entity models built using LSE, LSI, LDA and word2vec (§4.5) on *Home & Kitchen* and *Pet Supplies* product search benchmarks (§4.3) and evaluated on the test sets. The plots for *Clothing, Shoes & Jewelry* and *Sports & Outdoors* product search benchmarks are qualitatively similar to the ones shown. The figures show the absolute performance in terms of NDCG of $\tilde{e}_c^*$ (dashed curves) and $f(q)$ (solid curves); significance (§4.4) for the results for the approximately ideal retrieval vectors $\tilde{e}_c^*$ is computed between LSE and the best-performing baseline for each vector space size and indicated along the x-axis.

7. CONCLUSIONS

We have introduced Latent Semantic Entities, an unsupervised latent vector space model for product search. It jointly learns a unidirectional mapping between, and latent vector representations of, words and products. We have also defined a formalism for latent vector space models where latent models are decomposed into a mapping from word sequences to the product vector space, representations of products in that space, and a similarity function. We have evaluated our model using Amazon product data, and compared it to state-of-the-art latent vector space models for product ranking (LSI, LDA and word2vec). LSE outperforms all baselines for lower-dimensional vector spaces.

In an analysis of the vector space models, we have compared the performance achieved with the ideal performance of the proposed product representations. We have shown that LSE constructs better product representations than any of the baselines. In addition, we have obtained important insights w.r.t. how much performance there is to gain by improving the individual components of latent vector space models. Future work can focus on improving the mapping from words to products by incorporating specialized features or increasing the mapping's complexity. In addition, semi-supervised learning may help specialize the vector space and mapping function for particular retrieval settings.

A comparison of LSE with a smoothed lexical language model unveils that the two methods make very different errors. Some directed product search queries require lexical matching, others benefit from the semantic matching capabilities of latent models. We have evaluated LSE as a feature in a machine-learned ranking setting and found that adding LSE to language models and popularity-based features significantly improves retrieval performance.

As to future work, in this paper we focus on the unsupervised setting where we have a description and a set of reviews associated with every product. Fig. 6 shows that there is a lot of performance to gain by improving the query projection function f. In a semi-supervised setting, the difference between e_c^* and $f(q)$ can be minimized according to pairs of queries and ideal rankings. As an additional step, query-relevance training data could be incorporated during estimation of the entity space E. Moreover, as mentioned in §6, the query projection function f is expected to learn a more complicated mapping. Hence, it may be beneficial to consider incorporating additional hierarchical depth using multiple non-linear transformations in the construction of f. More generally, the obtained product representations can be beneficial for various entity-oriented prediction tasks such as entity disambiguation or related entity finding. While we have focused on product retrieval in this work, the proposed model, insights and ideas can be applied in broader settings, such as entity finding and ad-hoc retrieval.

Acknowledgments. The authors would like to thank Artem Grotov, Nikos Voskarides, Zhaochun Ren, Tom Kenter, Manos Tsagkias, Hosein Azarbonyad and the anonymous reviewers for their valuable comments and suggestions. This research was supported by Ahold, Amsterdam Data Science, Blendle, the Bloomberg Research Grant program, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELIAS, the Google Faculty Research Award scheme, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project number 027.012.105, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, 652.002.-001, 612.001.551, 652.001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] K. Balog. On the investigation of similarity measures for product resolution. In *LHD workshop at IJCAI*, 2011.
[2] K. Balog and M. de Rijke. Determining expert profiles (with an application to expert finding). *IJCAI*, 7:2657–2662, 2007.
[3] K. Balog and R. Neumayer. A test collection for entity search in dbpedia. In *SIGIR*, pages 737–740. ACM, 2013.
[4] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, pages 43–50, 2006.
[5] K. Balog, P. Serdyukov, and A. P. de Vries. Overview of the TREC 2010 entity track. Techn. report, DTIC Document, 2011.
[6] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, and L. Si. Expertise retrieval. *Found. & Tr. in Inform. Retr.*, 6(2-3):127–256, 2012.
[7] M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, 2014.
[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003.
[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.
[10] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.
[11] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.
[12] R. Cai, H. Wang, and J. Zhang. Learning entity representation for named entity disambiguation. In *Chin. Comp. Ling. and Nat. Lang. Proc. Based on Nat. Ann. Big Data*, pages 267–278. Springer, 2015.
[13] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12(Aug):2493–2537, 2011.
[14] A. P. de Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *Focused Access to XML Documents*, pages 245–251. Springer, 2007.
[15] S. C. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6): 391–407, 1990.
[16] G. Demartini, J. Gaugaz, and W. Nejdl. A vector space model for ranking entities and its application to expert search. In *Advances in Information Retrieval*, pages 189–201. Springer, 2009.
[17] L. Deng, X. He, and J. Gao. Deep stacking networks for information retrieval. In *ICASSP*, pages 3153–3157, 2013.
[18] H. Duan and C. Zhai. Mining coordinated intent representation for entity search and recommendation. In *CIKM*, pages 333–342. ACM, 2015.

[19] H. Duan, C. Zhai, J. Cheng, and A. Gattani. A probabilistic mixture model for mining and analyzing product search log. In *CIKM*, pages 2179–2188. ACM, 2013.

[20] H. Duan, C. Zhai, J. Cheng, and A. Gattani. Supporting keyword search in product database: A probabilistic approach. *Proceedings of the VLDB Endowment*, 6(14):1786–1797, Sept. 2013.

[21] Y. Fang, L. Si, and A. P. Mathur. Discriminative models of integrating document evidence and document-candidate associations for expert search. In *SIGIR*, pages 683–690. ACM, 2010.

[22] I. Forrester Research. US online retail forecast, 2010 to 2015, February 2012.

[23] M. Gäde, M. Hall, H. Huurdeman, J. Kamps, M. Koolen, M. Skov, E. Toms, and D. Walsh. Overview of the SBS 2015 interactive track. In *CLEF 2015*. Springer, 2015.

[24] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[25] D. Graus, M. Tsagkias, W. Weerkamp, E. Meij, and M. de Rijke. Dynamic collective entity representations for entity ranking. In *WSDM*. ACM, 2016.

[26] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304, 2010.

[27] G. E. Hinton. Learning distributed representations of concepts. In *8th Ann. Conf. of the Cogn. Sci. Soc.*, page 12, Amherst, MA, 1986.

[28] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.

[29] P.-s. Huang, N. M. A. Urbana, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338, 2013.

[30] B. J. Jansen and P. R. Molina. The effectiveness of web search engines for retrieving relevant ecommerce links. *Information Processing & Management*, 42(4):1075–1098, 2006.

[31] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142. ACM, 2002.

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[33] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, pages 595–603, 2014.

[34] H. Li and J. Xu. Semantic matching in search. *Found. & Tr. in Information Retrieval*, 7(5):343–469, June 2014.

[35] S. Liang and M. de Rijke. Formal language models for finding groups of experts. *Information Processing & Management*, 2016.

[36] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[37] X. Liu, W. B. Croft, and M. Koll. Finding experts in community-based question-answering services. In *CIKM*, pages 315–316. ACM, 2005.

[38] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *KDD*, pages 785–794. ACM, 2015.

[39] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

[40] S. McPartlin, L. F. Dugal, M. Jenson, and I. W. Kahn. Understanding how US online shoppers are reshaping the retail experience. PricewaterhouseCoopers, 2012.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[42] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient estimation of word representations in vector space. arXiv 1301.3781, 2013.

[43] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *ICML*, pages 641–648, 2007.

[44] A. Mnih and G. Hinton. A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088, 2008.

[45] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, pages 2265–2273, 2013.

[46] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *ICML*, pages 1751–1758, 2012.

[47] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848, 2002.

[48] P. Nurmi, E. Lagerspetz, W. Buntine, P. Floréen, and J. Kukkonen. Product retrieval for grocery stores. In *SIGIR*, pages 781–782. ACM, 2008.

[49] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543, 2014.

[50] L. Quoc and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[51] J. Rowley. Product search in e-shopping: a review and research propositions. *Journal of Consumer Marketing*, 17(1):20–35, 2000.

[52] R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approximate Reasoning*, 50(7):969–978, 2009.

[53] D. Sculley. Large scale learning to rank. In *NIPS Workshop on Advances in Ranking*, 2009.

[54] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*, pages 101–110, 2014.

[55] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, pages 623–632. ACM, 2007.

[56] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394. Association for Computational Linguistics, 2010.

[57] C. Van Gysel, M. de Rijke, and M. Worring. Unsupervised, efficient and semantic expertise retrieval. In *WWW*, 2016.

[58] V. Vapnik. *Statistical Learning Theory*, volume 1. Wiley New York, 1998.

[59] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[60] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *TOIS*, 22(2):179–214, 2004.

[61] Y. Zhao, L. Zhiyuan, and M. Sun. Representation learning for measuring entity relatedness with rich information. In *IJCAI*, 2015.

APPENDIX

Denote $p_k = \tilde{P}(x_k \mid w_{k,1}, \ldots, w_{k,n})$. The derivative of (5) w.r.t. bias term b equals

$$\frac{\partial L}{\partial b}(W_v, W_e, W, b) = -\frac{1}{m}\left(\sum_{k=1}^{m} \frac{1}{p_k}\frac{\partial p_k}{\partial b}\right)$$

and w.r.t. an arbitrary matrix parameter θ (W_v, W_e or W):

$$\frac{\partial L}{\partial \theta}(W_v, W_e, W, b) = -\frac{1}{m}\left(\sum_{k=1}^{m} \frac{1}{p_k}\frac{\partial p_k}{\partial \theta}\right) + \frac{\lambda}{m}\sum_{i,j}\theta_{i,j}.$$

Ignoring the subscripts for batch instances and word positions for ease of notation, for a single instance we denote x^+ as the target entity and X^- as the sample of z contrastive negative examples. We have

$$p \;=\; P(\mathcal{S} \mid e^+, f(w_{j,1}, \ldots, w_{j,n})) \cdot \prod_{x^- \in X^-} (1 - P(\mathcal{S} \mid e^-, f(w_{j,1}, \ldots, w_{j,n})))$$

where application of the product rule in the computation of $\frac{\partial p}{\partial \theta}$ is omitted due to space constraints.

For θ (W, b, W_e or W_v) we observe

$$\frac{\partial P(\mathcal{S} \mid e, f(w_{j,1}, \ldots, w_{j,n}))}{\partial \theta} = P(\mathcal{S} \mid e, f(w_{j,1}, \ldots, w_{j,n})) \cdot (1 - P(\mathcal{S} \mid e, f(w_{j,1}, \ldots, w_{j,n}))) \cdot \frac{\partial\, e \cdot f(w_1, \ldots, w_n)}{\partial \theta}$$

For a single entity representation e (a row of matrix W_e),

$$\frac{\partial\, e \cdot f(w_1, \ldots, w_n)}{\partial e} = f(w_1, \ldots, w_n)$$

where we observe that the update to an entity representation is the projected representation of the input n-gram multiplied by a scalar.

The symbolic derivative of the dot product between the entity representation and the projected n-gram w.r.t. bias term b, linear map W and word representations W_v, respectively, are:

$$\frac{\partial\, e \cdot f(w_1, \ldots, w_n)}{\partial b} = e \odot \operatorname{sech}^2\left(W \cdot (W_v \cdot \frac{1}{|s|}\sum_{w_i \in s} \delta_i) + b\right)$$

$$\frac{\partial\, e \cdot f(w_1, \ldots, w_n)}{\partial W} = \left(e \odot \operatorname{sech}^2\left(W \cdot (W_v \cdot \frac{1}{|s|}\sum_{w_i \in s} \delta_i) + b\right)\right) \cdot \left(W_v \cdot \frac{1}{|s|}\sum_{w_i \in s} \delta_i\right)^\intercal$$

$$\frac{\partial\, e \cdot f(w_1, \ldots, w_n)}{\partial W_v} = W^\intercal \cdot \left(e \odot \operatorname{sech}^2\left(W \cdot (W_v \cdot \frac{1}{|s|}\sum_{w_i \in s} \delta_i) + b\right)\right) \cdot \left(\frac{1}{|s|}\sum_{w_i \in s} \delta_i\right)^\intercal$$