Mix 'n Match: Integrating Text Matching and Product Substitutability within Product Search

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

DOI
10.1145/3269206.3271668

Publication date
2018

Document Version
Final published version

Published in
CIKM '18

Mix 'n Match: Integrating Text Matching and Product Substitutability within Product Search

Christophe Van Gysel*
Apple Inc.
cvangysel@apple.com

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

Evangelos Kanoulas
University of Amsterdam
e.kanoulas@uva.nl

ABSTRACT

Two products are substitutes if both can satisfy the same consumer need. Intrinsic incorporation of product substitutability—where substitutability is integrated within latent vector space models—is in contrast to the extrinsic re-ranking of result lists. The fusion of text matching and product substitutability objectives allows latent vector space models to mix and match regularities contained within text descriptions and substitution relations. We introduce a method for intrinsically incorporating product substitutability within latent vector space models for product search that are estimated using gradient descent; it integrates flawlessly with state-of-the-art vector space models. We compare our method to existing methods for incorporating structural entity relations, where product substitutability is incorporated extrinsically by re-ranking. Our method outperforms the best extrinsic method on four benchmarks. We investigate the effect of different levels of text matching and product similarity objectives, and provide an analysis of the effect of incorporating product substitutability on product search ranking diversity. Incorporating product substitutability information improves search relevance at the cost of diversity.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**;

KEYWORDS

Latent vector space models, Entity similarity, Product search

ACM Reference Format:
Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2018. Mix 'n Match: Integrating Text Matching and Product Substitutability within Product Search. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271668

1 INTRODUCTION

Revenue from online retail is growing with an average of 15% on a yearly basis [47]. Online retail has become a fundamental part of society [31]. Compared to brick-and-mortar stores, online stores typically carry many more products as physical aisles and promotional posters are replaced by their virtual counterparts: ranked lists and banner ads. E-commerce search engines are a crucial component of web stores that allow online shoppers to effortlessly navigate the extensive product catalogs [18]. To navigate the large numbers of available products, users formulate e-commerce search queries using traits of the product they are interested in [38]. When ranking products according to relevance w.r.t. the user's query, (product) search engines combine different signals in a learning to rank framework [40, 46]. The signals include, among others, the degree of text matching between the query and different product/document fields or attributes, and product-specific attributes, such as sales rating or price. One important source of product relevance that is often overlooked, however, is *product substitutability*. Henderson and Quandt [16, p. 29] state that "two goods (i.e., products) are substitutes if both can satisfy the same need to the consumer" [23]. Consequently, the question we address in this paper is whether the incorporation of product entity substitutability can improve the effectiveness of relevance ranking within product search.

The substitutability relation among products e_a and e_b can be determined in multiple ways. Cross-price elasticity, the change in demand for product e_a w.r.t. the change in price of product e_b [23], is a method from economics that measures the relation between pairs of products [15]. A positive cross-price elasticity indicates that raising the price of product e_b, all else unchanged, caused a rise in the demand for product e_a and vice versa for price decreases. Given that the world is dynamic and thus cross-price elasticity is hard to estimate, Lattin and McAlister [23] proposed alternative methods where products are represented by a set of features. Another source of information regarding product relations is the *Commodity and Food Elasticities* database of the U.S. government [35]. However, on e-commerce platforms, product substitutability can be inferred from user purchasing behavior [30]. For example, if multiple users view product e_a before purchasing product e_b, then the substitutability of product e_a and e_b can be determined empirically.

The key idea of this paper is that product substitutability relations can facilitate the retrieval of relevant products that are impacted most by the **vocabulary gap** [25]. That is, we postulate that product substitutability relations can help bring products—with textual representations that do not match very well with the query—up higher in the ranking. Product substitutability can be integrated into product search either extrinsically or intrinsically. *Extrinsic* methods perform a re-ranking of text retrieval model rankings [45]. Consequently, these methods can be applied to any existing retrieval model that generates a ranking of products. However, extrinsic methods are unable to use regularities shared between text and product substitutability to their advantage as the text matching component is abstracted away by the underlying retrieval model. That is, if products e_a and e_b are deemed substitutable and the textual representation of product e_b is semantically similar to that of product e_c, then existing external methods fail to recognize the similarity between products e_a and e_c that arises due to the transitive

*Work performed while at the University of Amsterdam.

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271668

relation that operates across text descriptions and product substitutability. With *intrinsic* incorporation, product substitutability relations are embedded within the retrieval models. Consequently, the retrieval models can take into account signals from both text matching and product substitutability to improve product search effectiveness. The framework we present in this paper directly and intrinsically incorporates entity similarity within state-of-the-art latent vector space models [48, 53] that are learned using gradient descent. Our approach has the following benefits. First, and to address the limitation of existing work we raised previously, the fusion of a text matching objective and our novel entity similarity objective allows the latent vector space model to mix 'n match both signals within the vector space. Consequently, transitive similarity across text and substitutability is modeled. Second, as the incorporation of entity similarity relations occurs during model estimation only, few changes need to be made to existing production search engines based on latent vector space retrieval in order to benefit from improved effectiveness with only a slight increase in model training time. This is contrary to a re-ranking approach that requires an additional step to be implemented in the retrieval pipeline.

We answer the following research questions: (1) How does the intrinsic incorporation of product similarity into latent vector space models compare to the extrinsic combination of text matching and product similarity signals in terms of retrieval effectiveness? (2) What is the effect of mixing different levels of the product similarity and the text matching signals within latent vector space models? (3) Can we explain what the incorporation of entity similarity contributes to the latent vector space models compared to a vector space that was constructed using only text?

We contribute: (1) A framework to integrate entity similarity signals within state-of-the-art neural vector space models, with an application to product search. (2) An extension to the C++/CUDA implementation of LSE and NVSM that allows one to incorporate entity similarity while training latent vector space models.[1] (3) Comparisons of extrinsic and intrinsic methods to incorporate entity similarity using various retrieval models (BM25, QLM, word2vec, LSE and NVSM). (4) Insight in the effect of mixing text matching and entity similarity on retrieval effectiveness within latent vector space models. (5) A better understanding of domain-specific regularities within product departments. (6) Analysis of the effect of incorporating entity similarity on the product search rankings.

2 RELATED WORK

2.1 Product search

Users often navigate e-commerce websites through product search engines [18]. Santu et al. [40] note that product search requires specialized solutions and discuss practical challenges that arise when learning to rank products. Nurmi et al. [34] introduce a system that ranks products w.r.t. a natural language grocery list. Duan et al. [13] focus on the structured aspects of product entities and propose a mixture model for attribute-level analysis of search logs in e-commerce websites. Later, Duan et al. [12] extend their approach to enable filtering based on product attributes. Vandic et al. [55] propose several algorithms for automatic facet selection where they aim to minimize the number of facet selection steps needed to find a desired product. Duan and Zhai [11] learn a query intent representation for structured product entities. McAuley et al. [30] learn to recommend complementary and alternative products based on image similarity features and infer a network of substitutable/complementary products [29]. Van Gysel et al. [48] learn unsupervised latent product representations from unstructured product descriptions and customer reviews on Amazon data [26, 29, 30]. Ai et al. [3] introduce a supervised approach to learn personalized latent representations of products, users and queries.

Our work differs from the works listed above in that it incorporates product substitutability relations to improve relevance ranking within latent vector space models of products, contrary to models that incorporate text matching only [48] or personalize retrieval based on historical user interactions [3]. Compared to learning to rank e-commerce systems [3, 40], we make use of relations amongst products instead of query/product relevance pairs.

2.2 Incorporating entity similarity to improve retrieval effectiveness

Incorporating similarity between retrievable objects, such as entities or documents, is not a new idea [7]. The *cluster hypothesis* states that "closely associated documents tend to be relevant to the same requests" [54]. Product substitutability and the cluster hypothesis are related, but the latter is subordinate to the former. Typical applications of the cluster hypothesis [20] cluster documents based on, sometimes dimensionality-reduced [14], bag-of-word features. Retrieval effectiveness can then be improved by performing retrieval on the cluster level [21] and inducing an importance prior amongst documents [22], among others. In the context of product search, the difference between product substitutability and the cluster hypothesis lies in the directionality of both statements. That is, while the cluster hypothesis assumes that similar documents will have similar relevance w.r.t. an information need, product substitutability mandates that two goods that can satisfy the same consumption need are similar. Consequently, the cluster hypothesis can serve as a way to infer similarity between entities, whereas product substitutability assigns explicit semantics to pairs of similar products. Szummer and Yilmaz [44] incorporate textual similarity when learning to rank documents. In this paper, however, we take a different approach where we wish to incorporate prior knowledge of entity (i.e., product) similarity, which is not based on textual similarity, to improve product search effectiveness. Raviv et al. [36] show that the clustering hypothesis can hold for entity retrieval up to a substantial extent. More closely related to this paper, Tonon et al. [45] combine text matching and structured search over a knowledge graph to improve entity search effectiveness.

Parallel to information retrieval, there has been significant work with regard to the fusion of multi-modal information sources in multimedia [4]. In particular, Snoek et al. [42] compare *early* and *late* fusion of information sources for the task of semantic concept detection in videos and find that late fusion generally performs better. Compared to our work, early and late fusion correspond to intrinsic and extrinsic incorporation of product substitutability, respectively. Contrary to Snoek et al. [42] in the context of multimedia, we find that early fusion performs better for product search when fusing text matching and product substitutability.

2.3 Latent representations for information retrieval

Semantic matching, provided by latent semantic vector space models, is needed to bridge the vocabulary gap [25]. Latent semantic

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

spaces gained attention with the introduction of Latent Semantic Indexing [10] and its probabilistic variants [17]. Blei et al. [6] introduce Latent Dirichlet Allocation, an unsupervised probabilistic topic model that generalizes to documents unseen during training. Salakhutdinov and Hinton [39] introduce semantic hashing for finding similar documents. Mikolov et al. [32] introduce word2vec, a popular set of models for learning latent word representations based on a language modeling objective. Vulić and Moens [56] propose a methodology for using word-based representations to construct query and document representations for vector space retrieval. Kenter and de Rijke [19] employ word2vec representations to determine whether two short texts are paraphrases of each other (i.e., short text similarity rather than full text matching). Le and Mikolov [24] introduce doc2vec, an extension to word2vec where a representation for the document is learned as well. Ai et al. [2] integrate the doc2vec representations within a language model for information retrieval, but note several issues with the representations learned using doc2vec [1]. Van Gysel et al. [48, 49] introduce a representation learning model for product finding and they later extended it to ad-hoc retrieval [53]. Ai et al. [3] personalize the product vector space model so that, besides representations for queries and products, representations for users are learned as well.

Our proposed approach differs from the works listed above as it is complementary to document/entity representation models that are learned from scratch using gradient descent. We are the first to integrate entity relations within the estimation of latent vector space models for text-based entity ranking. The core idea here is that relations amongst entities can be used to push entities with little associated text closer to their related text-rich counterparts. More specifically, this allows low-text entities to piggy back on the rich textual relation learned for the text-rich entities that these low-text entities are related to. Moreover, our framework integrates flawlessly with existing latent vector space models for information retrieval [3, 48, 50, 52, 53] and improves their retrieval effectiveness as we demonstrate in the results section (§5).

3 INCORPORATING PRODUCT SIMILARITY

First, we discuss existing work that consists of *extrinsic* methods to incorporate product similarity signals that use retrieval models to obtain an initial product ranking and then re-rank the results by taking into account product similarity. We then proceed with an *intrinsic* incorporation of product similarity—the contribution of this paper—into the optimization objective of state-of-the-art latent vector space models for information retrieval.

3.1 Background

Viewed abstractly, the focus of this paper is on a ranking scenario where we wish to rank a set of entities $\mathcal{E}$ in accordance to a user-issued unstructured textual query q consisting of $|q|$ terms. In e-commerce search, entities are products where every product is characterized by an unstructured body of text [48] (title, description and reviews). On top of that, we consider an additional signal of entity similarity that indicates that one entity is a semantic alternative for another entity. In the case of general-purpose entity knowledge bases (such as Wikipedia), these relations could be the `<owl:sameAs>` [5, 5.2.1] or `<dbpedia:redirect>` [57] edges in the knowledge graph that indicate semantic identity between entities. In e-commerce search, the focus of this paper, products can be *substitutes* (e.g., pairs of shoes) or *complements* (e.g., tooth paste and a tooth brush) of each other [28, p.184]. These notions of similarity are captured in product relations based on user interaction [30]. For example, in the case of the popular e-commerce website Amazon.com [26], product substitutability [30] is characterized by two relations: (a) users who viewed product X also viewed product Y, and (b) users who viewed product X eventually bought product Y. Consequently, the main question we ask in this paper is whether the incorporation of product substitutability (i.e., the product similarity signal explored in this paper) can improve product retrieval effectiveness.

3.2 Extrinsic re-ranking using product similarity

Hybrid retrieval approaches that combine text matching and structured search have been explored before in the context of linked data. Tonon et al. [45] propose an approach that combines text retrieval models and the relations in a semi-structured database as follows. Let $f_R(q, e)$ denote the text matching score between query q and entity $e \in \mathcal{E}$ according to retrieval model R. In the original description, the retrieval model R was fixed to be the Okapi BM25 [37] retrieval model. However, in this paper we take a more general approach and assume that any retrieval model can take its place, such as probabilistic language models [58] or latent vector space models [48]. An initial ranking r of entities $\mathcal{E}$ for query q is obtained according to retrieval model R, and the top-k ranked entities are used to determine a set of entities that are related to the initial entities through similarity relations. The extrinsic re-ranking approach intuitively works as follows. For a particular query, we move through the top-k products in the result list from top to bottom. At every rank position i, we consider all products e that are alternatives for the product $e_{\text{rank}(i)}$ at the current rank as candidate products. Consequently, product $e_{\text{rank}(i)}$ is the product that leads to the discovery of product e, if e was not ranked higher than $e_{\text{rank}(i)}$ in the same result list. We then rank all products in the original ranking and the newly-discovered candidate products based on a retrieval score based on both products $e_{\text{rank}(i)}$ and e. More formally, we use S_e to denote the set of entities related to entity e; the seed entity $s(e)$ that leads to the discovery of entity e is defined as

$$s(e) = \arg\min_{e' \in C_e} \text{rank}\left(r, e'\right) \tag{1}$$

where $\text{rank}(r, e)$ denotes the rank of entity e in ranked list r and $C_e = \{e' \in \mathcal{E} \mid (e \in S_{e'} \wedge \text{rank}(r, e') \leq k) \vee e' = e\}$ is the set of candidate entities that can lead to the discovery of entity e. That is, the highest-ranked entity in the top-k that is similar to entity e or, if no entity in the top-k is similar to entity e, the entity e itself is the entity that leads to the discovery of entity e.

A final ranking r' of entities $e \in \mathcal{E}$ is then obtained by linearly combining the retrieval score of the entity $s(e)$ that led to the discovery of entity e and a re-scoring function $g(q, e, e')$ that quantifies the similarity between entities e and e' in accordance to query q:

$$f_{\text{inter}}(q, e) = \begin{cases} \lambda \cdot f_R(q, s(e)) + (1 - \lambda) \cdot g(q, e, s(e)), & \text{if } f_R(q, s(e)) \geq \gamma \\ -\infty, & \text{otherwise} \end{cases} \tag{2}$$

where λ is an interpolation hyperparameter (§4.3.1) that indicates the amount of structured expansion to be used and γ is a threshold (§4.3.1) on the retrieval score of seed entity $s(e)$ that led to the

discovery of entity e that filters out noise. The desired ranking r' is then obtained by sorting entities in decreasing order of $f_{\text{inter}}(q, e)$.

Next, the re-scoring function $g(q, e, e')$ can take several forms. In the case of Tonon et al. [45], $g(q, e, e')$ is an aggregation over the retrieval score between different entity fields and the query. In this paper, every entity is represented as an unstructured body of text. Consequently, we adapt the definition of Tonon et al. [45] to our setting by introducing *structural* and *text matching* variants of $g(q, e, e')$. In the structural definition, the function $g_{\text{struct}}(q, e, e')$ is independent of the discovered entity:

$$g_{\text{struct}}(q, e, e') = f_R(q, e') - \epsilon, \tag{3}$$

where ϵ is a hyperparameter (§4.3.1) that allows for adjusting the retrieval model score. Using the definition of Eq. 3 in combination with Eq. 2, discovered entities are ranked at least as high as the seed entity that led to their discovery. This makes sense if the entity discovered through structured expansion has only little associated text. The text matching variant of $g(q, e, e')$ provides an alternative:

$$g_{\text{match}}(q, e, e') = f_R(q, e) - \epsilon, \tag{4}$$

where ϵ has the same semantics as in Eq. 3. Using the definition of Eq. 4, the final retrieval score becomes a weighted average of the retrieval scores of both entities (when $\epsilon = 0$). For both variants of $g(q, e, e')$ defined in Eq. 3 and 4, the final ranking remains the same as the initial ranking if no related entities are discovered.

The hybrid retrieval method originally introduced by Tonon et al. [45] is targeted at a retrieval setting where structured entities are characterized by a large number of fields with limited text, contrary to the unstructured setting we consider in this paper. The method just introduced differs from that of Tonon et al. [45] as follows: (1) In [45], the initial retrieval model was limited to BM25 only; we do not impose this limitation. (2) Tonon et al. [45] considered Jaro-Winkler (JW) distance to perform text matching on structured entity attribute fields in addition to BM25; they reported that the structured variant using BM25 (Eq. 3) performed best and consequently we do not consider JW distance as it is not typically applied to unstructured text retrieval. (3) Finally, the text matching variant g_{match} (Eq. 4) was introduced as part of this paper to accommodate unstructured text and thus is a novel contribution over [45].

Both definitions of g (i.e., g_{struct} and g_{match}) make use of substitutability relations as $g(q, e, e')$ is evaluated only if e is a substitute for e' and vice versa. Their difference, however, lies in how the substitutability relation is converted into a score that boosts the low-ranked substitute to a higher position in the ranked list. For the explanation that follows, we assume that product e' is ranked highly and its substitute, product e, suffers from the vocabulary gap and consequently is incorrectly ranked low in the returned product ranking. The structural variant of g, g_{struct}, uses the retrieval score of the product that is already ranked highly, product e', as a proxy for the retrieval score of the low-ranked substitute, product e. This variant makes sense if the textual representation of the lower-ranked substitute, product e, does not match the query at all. The retrieval score of higher-ranked product e' is then used to improve the ranking by assuming that the textual representation of the highly-ranked product e' is also valid for its lower-ranked substitute e. The matching variant of g, g_{match}, does not completely ignore the retrieval score of the lower-ranked substitute product e (as is the case with g_{struct}), and instead imposes the assumption that the lower-ranked substitute product e matches the query somewhat—but not very well. Thus, the integration of g_{match} (Eq. 4) within f_{inter} (Eq. 2) makes the final retrieval score of the lower-ranked substitute a combination of the retrieval score of the highly-ranked product e' and the retrieval score of the low-ranked substitute product e itself.

3.3 Incorporating product similarity within latent vector space models

Recent advances in entity ranking led to the introduction of latent vector space models that learn a relation between words and entities using batched gradient descent. Van Gysel et al. introduced several methods for estimating latent vector space models for tasks ranging from entity ranking to document retrieval [48, 49, 53]. The current body of work focuses solely on unsupervised learning of the word-entity relation and does not take into account additional sources of relevance. Therefore, in this paper, we extend the loss function of these latent vector space models such that they *intrinsically* incorporate similarity between entities. In contrast to the *extrinsic* method described in §3.2, where existing rankings are augmented with structured information, the incorporation of structured information into latent vector space models occurs directly during parameter estimation. Thus, retrieving entities using latent vector space models that incorporate entity similarity remains a matter of ranking entities according to the nearest neighbors of a query vector based on the user-issued query q.

We present here a generic framework to incorporate entity similarity based on the observation that, within the current latent vector space models that are learned using gradient descent [48, 53], the loss function is a weighted sum of sub-loss functions that incorporate *text matching* and *regularization* as follows:

$$L(\theta \mid D_{\text{text}}) = L_{\text{text}}(\theta \mid D_{\text{text}}) + \lambda_{\text{reg}} \cdot L_{\text{reg}}(\theta), \tag{5}$$

where θ is the set of parameters of the latent vector space, D_{text} is the training data from which the model is estimated and λ_{reg} is a hyperparameter (§4.3.1) controlling the amount of regularization. For the text objective L_{text}, the data D_{text} takes the form of pairs consisting of an entity and an n-gram extracted from the entity's description; we refer to [48, 53] for details. Regularization loss L_{reg} is usually the l2 regularization loss (i.e., the sum of squares of individual parameters [33]). Note that Eq. 5 is optimized using only the n-grams that occur in the textual content of entities (i.e., product title, description and reviews in this paper) and does not make use of query/product relevance information (e.g., clicks, manual relevance judgments).

We extend the loss function of Eq. 5 by adding an additional sub-loss function that specifies that entities that are similar (i.e., entities e_i, e_j where $e_i \in S_{e_j}$ and vice versa) should be nearby in latent vector space. More specifically, we have

$$\begin{aligned} L(\theta \mid D_{\text{text}}, D_{\text{sim}}) \quad = \quad & \alpha \cdot L_{\text{text}}(\theta \mid D_{\text{text}}) \\ & + (1 - \alpha) \cdot L_{\text{sim}}(\theta \mid D_{\text{sim}}) \\ & + \lambda_{\text{reg}} \cdot L_{\text{reg}}(\theta), \end{aligned} \tag{6}$$

where L_{sim} denotes the sub-loss function that we will define next, $0 \leq \alpha < 1$ is a hyperparameter (§4.3.1) controlling the trade-off between the text matching and entity similarity objectives, and D_{sim} is a set of similar entity pairs.

The sub-loss function incorporating entity similarity is defined as follows. We write

$$\vec{R_{\mathcal{E}}}^{(i)}$$

to denote the k_e-dimensional latent vector representation of entity e_i (i.e., a subset of the learned parameters θ of the latent vector space model); then, we have

$$L_{\text{sim}}(\theta \mid D_{\text{sim}}) = -\sum_{(e_i,e_j)\in D_{\text{sim}}} \log \sigma\left(\vec{R_{\mathcal{E}}}^{(i)} \cdot \vec{R_{\mathcal{E}}}^{(j)}\right), \quad (7)$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ denotes the sigmoid function. Thus, the representations of entities that are similar are optimized to be nearby in the latent vector space. Eq. 6 is then optimized using batched gradient descent; see [48, 53] for details. Note that Eq. 6 only incorporates information of similar entities, but ignores entities that are not similar. If we would rely only on the entity similarity sub-objective (Eq. 7), then a trivial solution is to learn all entity representations to be equal to the null vector. However, given that we defined $0 \leq \alpha < 1$, the text matching sub-objective enforces that entities that have little semantic overlap in their textual description will have representations that are further apart than entities with semantically similar texts. Again, we emphasize here that no query/product relevance labels or data about user interaction are used during parameter estimation, but instead the combined loss function (Eq. 6) learns a matching function from (1) n-grams that occur within the textual representations of products (i.e., Eq. 5 as defined in [48, 53]), and (2) product substitutability relations (i.e., the contribution of this paper).

Over and above existing work on semantic product search [3, 48], we have just introduced a mechanism to incorporate product substitutability relations within latent vector space models. Two products are substitutes if they satisfy the same need for the consumer. Consequently, the incorporation of product substitutability within retrieval models is expected to improve their effectiveness as information and consumption needs are two sides of the same coin within the product search domain.

In §1 we emphasized the importance of effectively utilizing the cross-modal similarity relations that are established by textual semantics and product substitutability. More specifically, if products e_a and e_b are deemed substitutable and the textual representation of product e_b is semantically similar to that of product e_c, then products e_a and e_c are also similar. This transitivity relation is conveniently integrated within the latent vector space models that are the subject of this section as follows. First, within a latent vector space model, similarity amongst entities is defined as the cosine similarity between their vector representations [53]. Cosine similarity corresponds to the chordal length between normalized vectors. Denote ϕ as the angle between vectors $u, v \in \mathbb{R}^n$, the chordal length $\text{crd}(\phi) = 2\cos\left(\frac{\pi-\phi}{2}\right)$ is equal to the Euclidean distance between the normalized vectors $\hat{u}, \hat{v}$. In particular, the smaller the chordal length between u, v, the greater their cosine similarity, $\cos(\phi)$, will be. Imagine now a third vector, $w \in \mathbb{R}^n$, and the chordal lengths between u, v and w (i.e., $\|\hat{u} - \hat{w}\|$ and $\|\hat{v} - \hat{w}\|$). The triangle inequality tells us that $\|(\hat{u} - \hat{w}) + (\hat{v} - \hat{w})\| = \|\hat{u} - \hat{v}\| \leq \|\hat{u} - \hat{w}\| + \|\hat{v} - \hat{w}\|$ (note here that we made use of the symmetry of Euclidean distance, i.e. $\|\hat{v} - \hat{w}\| = \|\hat{w} - \hat{v}\|$). Consequently, if u, w and v, w are similar, then u, v will also be similar and thus cosine similarity is transitive. Secondly, the loss function in Eq. 6 is an additive mixture of cross-modal sub-objectives that encode text/product and product/product similarity within the model's representations. Consequently, textual and entity similarity (i.e., product substitutability in this paper) information will be combined in the resulting model as to best optimize the loss (Eq. 6). Therefore, if products e_a, e_b are substitutes and the texts of products e_b, e_c are semantically similar, then products e_a, e_c will have similar representations as long as their textual semantics do not differ drastically (trade-off hyperparameter α above controls what is deemed drastic).

4 EXPERIMENTAL SETUP

4.1 Research questions

In this paper we investigate whether the incorporation of product similarity signals can benefit product retrieval effectiveness. We seek to answer the following research questions:

RQ1 How does the intrinsic incorporation of product similarity into latent vector space models compare to the extrinsic combination of text matching and product similarity signals in terms of retrieval effectiveness?

In particular, does the intrinsic incorporation (§3.3) of these signals yield a significant improvement over the extrinsic approaches (§3.2) for latent vector space models? What is the impact of applying the extrinsic re-ranking methods to retrieval models where the intrinsic incorporation of similarity signals (e.g., BM25 and non-neural latent vector space models) does not come naturally?

RQ2 What is the effect of mixing different levels of the product similarity and the text matching signals within latent vector space models?

Within latent vector space models, the entity similarity objective is auxiliary to the text matching objective. After all, a ranking is obtained w.r.t. a textual query and thus the query cannot be taken into account if only entity similarity is learned. How important is the entity similarity signal and what is the effect of different levels of mixing it with the text matching objective on retrieval effectiveness?

RQ3 Can we explain what the incorporation of entity similarity contributes to the latent vector space models compared to a vector space that was constructed using only text?

If the incorporation of entity similarity improves retrieval effectiveness, can we explain *why*? Can we find any recurring patterns when we examine the set of newly-discovered relevant documents that are ranked highly when product similarity is taken into account, but are ranked low otherwise?

4.2 Product search benchmarks & experiments

To answer our research questions regarding the potential of entity similarity to improve retrieval effectiveness, we follow the experimental setup of [3, 48], where Amazon products are represented by their title, description and reviews from Amazon customers.

4.2.1 Benchmarks. We evaluate the extrinsic and intrinsic methods (§3) using various retrieval models (§4.3) on product domains of increasing size: Pet Supplies (32,768 products), Sports & Outdoors (65,536 products), Toys & Games (131,072 products) and Electronics (262,144 products); see Table 1 for an overview. The two smaller benchmarks (Pet Supplies, Sports & Outdoors) are identical to the largest benchmarks released as part of [48]. To show that our findings scale to retrieval settings of ∼100k products and more, we constructed two larger benchmarks (Toys & Games, Electronics) in exactly the same way as [48] for this paper. That is, products are represented by the concatenation of their title, description and user

Table 1: Overview of the product search benchmarks. T and V denote the test and validation sets, respectively. Mean and standard deviation are reported wherever applicable. Product text length statistics are computed on the concatenated textual title, description and customer reviews associated with the products. The two smaller collections (Pet Supplies, Sports & Outdoors) are the ones released as part of [48], whereas the larger collections (Toys & Games, Electronics) were constructed in the same fashion as [48] for the purpose of this paper (see §4.2.1 for details).

		Pet Supplies	Sports & Outdoors	Toys & Games	Electronics
Collection (training)					
Products		32,768	65,536	131,072	262,144
Product text length		986.03 ± 5,116.09	555.85 ± 2,549.72	542.16 ± 2,076.78	1,541.15 ± 8,518.67
Unique terms		2.75×10^5	3.94×10^5	5.67×10^5	2.23×10^6
Substitutes per product		7.69 ± 14.53	2.62 ± 6.29	13.28 ± 21.42	8.46 ± 18.99
Queries and relevance (testing)					
Queries	(T)	385	1,879	394	601
	(V)	42	208	98	150
Query terms		4.73 ± 1.62	5.64 ± 1.68	6.00 ± 2.47	5.55 ± 1.55
Relevant products	(T)	75.96 ± 194.44	29.27 ± 61.71	258.92 ± 1,041.68	324.26 ± 776.65
	(V)	57.40 ± 88.91	38.25 ± 157.34	188.56 ± 493.11	412.85 ± 1,414.43

reviews. Specific to this paper, product substitutability relations are obtained from user interaction behaviour as first introduced by McAuley et al. [30]. The product substitutability relations are then used as the product similarity signal in this paper. More specifically, on Amazon.com, product substitutability between product X and product Y is characterized by two relations: (a) users who viewed product X also viewed product Y, and (b) users who viewed product X eventually bought product Y [30]. These relations are extracted from user interaction logs by counting interaction co-occurrences as described in detail by Linden et al. [26, algorithm on p. 79].

Rowley [38, 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." Based on this observation [3, 48], test queries are extracted from category hierarchies and query/product relevance is determined based on category membership. Category hierarchies of less than two levels are ignored, as the first level in the category hierarchy is often non-descriptive for the product. Products can be relevant for multiple queries. Textual queries are extracted from a category hierarchy by concatenating the names of the hierarchy levels and removing duplicate and stop words. We refer to [3, 48] for more details and examples. Note that in this paper—similar to [48], but unlike [3]—queries and their relevant products are used for hyperparameter selection (§4.3.1) and testing only. None of the methods in this paper make use of any training queries or relevance labels to estimate their trainable parameters. For the existing collections (Pet Supplies, Sports & Outdoors) we use the validation and test splits as released as part of [48]. For the collections we constructed for the purpose of this paper (Toys & Games, Electronics), we randomly create a split of validation (20%) and test (80%) queries.[2]

4.2.2 Experimental design. We answer **RQ1** by comparing extrinsic and intrinsic methods to incorporate product similarity using lexical and semantic retrieval models (§4.3). In particular, we focus on the relative increase in retrieval effectiveness for each retrieval model individually when product similarity is incorporated. To address **RQ2**, we perform a parameter sensitivity analysis where we measure the effect of the interpolation hyperparameter α (Eq. 6) on retrieval effectiveness. For **RQ3**, we construct maximum-likelihood unsmoothed categorical language models [58] of the textual content associated with the top-100 products ranked by latent vector space models that are estimated using (a) text only and (b) text and product similarity signals. We then measure the per-query differences in ranking language perplexity for both language models, and consequently make inferences about the predictability and diversity of the language contained within the respective rankings.

4.3 Retrieval models & baselines for incorporating similarity

The product collection is indexed by Indri[3] [43] using `pyndri` [51]; every product is represented by a single document that consists of the concatenation of its title, description and customer reviews. Non-Indri retrieval models (i.e., latent vector space models) access the underlying tokenized document representations to ensure consistent tokenization. The hyperparameters mentioned below are optimized on the validation set (Table 1) according to MAP@1000.

4.3.1 Retrieval models. We evaluate the following retrieval models (based on the experimental setup of [53]): (1) Okapi BM25 [37] with the default Indri hyperparameter values ($k_1 = 1.2, b = 0.75$ and $k_3 = 7$), (2) QLM [58] with Dirichlet smoothing with hyperparemeter $\mu \in \{125, 250, 500, 750, 1000, 2000, 3000, 4000, 5000\}$ optimized on the validation set. (3) word2vec (w2v) [32] with Skip-Gram. We follow the methodology of Vulić and Moens [56], where query/document representations are constructed from the words contained within them by taking (a) the unweighted sum of the word representations (`add`), or (b) the sum of word representations weighted by the words' self-information (`si`), a quantity similar to Inverse Document Frequency (IDF) [9]. Vocabulary filtering and frequent word subsampling was disabled such that the input to all algorithms is the same. The word embedding size is set to 256. The one-sided window size ($\{x/2 \mid x = 4, 8, 16\}$) and the number of training epochs (between 1 and 15) is optimized on the validation set. All other parameters are configured with their default values (Gensim[4] implementation). (4) LSE [48] and NVSM [53] where we

[2] A list of product identifiers, queries, query/product relevance pairs and test/validation query splits is available at https://github.com/cvangysel/cuNVSM.

[3] The out-of-the-box Indri normalization was applied and the standard Indri stop word list was used to remove stop words.

[4] https://github.com/RaRe-Technologies/gensim

mostly follow default hyperparameter values; see [53]. The entity representation size is set to 256 [53, §6] and the batch size to 8,192. Similar to word2vec, the window size ($\{x \mid x = 4, 8, 16\}$) and number of training epochs (between 1 and 15) are optimized on the validation set. For word2vec, LSE and NVSM, the vocabulary is limited to the top-60k words to avoid data sparsity issues [48, 53].

4.3.2 Methods to incorporate similarity. We compare the extrinsic and intrinsic methods from §3: (1) For the extrinsic re-ranking methods we compare both variants based on structural (Eq. 3) and text matching (Eq. 4) information sources. The interpolation hyperparameter $\lambda = 0.1, 0.2, \ldots, 1.0$ (Eq. 2), the number of top-$k = 1, 3, 5, 10$ products used to discover similar products, score-adjusting hyperparameter ϵ and threshold γ are optimized on the validation set. For the score-adjusting hyperparameter ϵ and threshold γ, the set of values we consider for optimization is different for every retrieval model as their value ranges differ and 5 uniformly-spaced values are selected in the following ranges: (a) for the log-probability of QLM, $-10 \leq x \leq 0$, (b) for the BM25 score, $0 \leq x \leq 9$, and (c) for the cosine similarity of word2vec, LSE and NVSM, $-1 \leq x \leq 1$. The above grid search over hyperparameters λ, k, γ and ϵ is performed by first selecting the per-model configuration that performs best on the validation set and then performing the 4-dimensional hyperparameter search for the extrinsic methods. (2) We incorporate product similarity intrinsically within the loss functions (Eq. 6) of LSE/NVSM, where $\alpha = 0.0, 0.1, \ldots, 1.0$ is optimized on the validation set together with all other hyperparameters of LSE/NVSM mentioned above as the entity similarity signal is integrated within the learning algorithm.

4.4 Evaluation measures & statistical significance

Mean Average Precision at rank 1000 (MAP@1000) is our principal evaluation measure that we use to address **RQ1**, **RQ2** and **RQ3**. For **RQ1**, we additionally report Normalized Discounted Cumulative Gain at rank 100 (NDCG@100) and precision at rank 10 (P@10). Evaluation measures are computed using the official tool released by TREC, `trec_eval`.[5] Significance of observed differences between the obtained results is determined using a two-tailed paired Student's t-test [41] ($^{***}p < 0.01$; $^{**}p < 0.05$; $^{*}p < 0.1$).

5 RESULTS

5.1 Extrinsic vs. intrinsic methods

RQ1 Table 2 shows the product retrieval effectiveness for various retrieval models (§4.3.1) using different methods to incorporate product similarity (§4.3.2).

We observe that the extrinsic g_{struct} method [45] does not perform well on the product search setting. This can be explained by the fact that the method was originally introduced for retrieving multi-fielded entities where text is scarce and relies only on the text matching score of entities part of the original ranking. In our setting, where products are represented as an unstructured body of text, it is sensible to rely on the text matching score of the entity that was discovered through similarity relations (g_{match}) instead.

For each of the lexical models (BM25, QLM) individually, we observe that the extrinsic method using g_{match} yields an increase in effectiveness over the text-only variants. The re-ranking of results using similarity relations likely incorporates a semantic matching signal within the result list. However, the increase in effectiveness using the extrinsic g_{match} quickly fades away if we examine the latent methods based on word2vec. Among the word2vec-based models, we observe a negligible increase (never more than $2 \cdot 10^{-3}$ difference in MAP@1000) in effectiveness when extrinsic expansion of the result list is used. The same observation (albeit with a maximum increase of $3 \cdot 10^{-3}$ in MAP@1000) holds for the use of the extrinsic method (g_{match}) when used to re-rank the neural latent vector space models (LSE, NVSM) results.

Examining the effectiveness of incorporating entity similarity intrinsically (Eq. 6) within latent vector space models, we observe disappointing results for LSE. More specifically, LSE is only able to outperform the extrinsic methods for 2 out of 4 benchmarks (Pet Supplies and Electronics). In the case of NVSM, however, we observe a significant ($p < 0.05$) increase in MAP@1000 on all product search benchmarks over the extrinsic re-ranking methods. The product substitutability signal that was added in Eq. 6 only pertains to the relation amongst products and thus only indirectly influences the relation between text and products (Eq. 5). While the text/product relation clearly benefits from the added substitutability relations—as can be seen in Table 2—this increase in performance is because the representations of substitutable products are pushed closer together in latent space, only if there is no strong textual signal that this should not be the case. Consequently, the difference in performance between LSE and NVSM can be explained due to the fact that LSE simply has inferior text matching capabilities compared to NVSM (as was already pointed out in [53] to be due to the lack of term specificity within LSE).

5.2 Trade-off between text and similarity

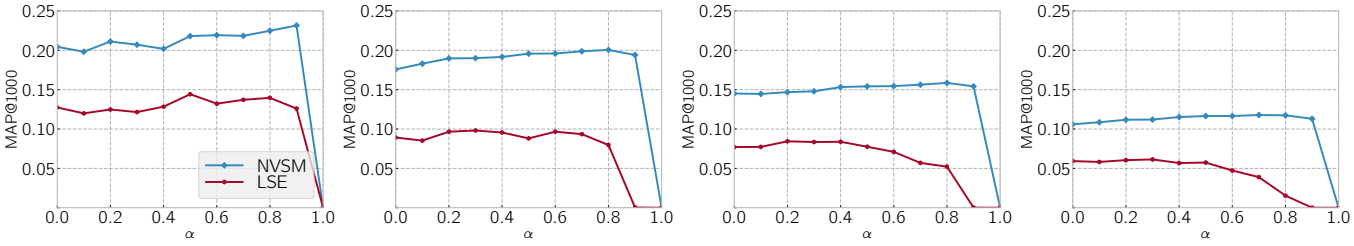
RQ2 The effect of the trade-off parameter α between text matching and product similarity signals within latent vector space models (§6) is shown in Fig. 1.

Different behavior is observed for both vector space models: while NVSM shows stable behavior w.r.t. the trade-off parameter α, and generally reaches a maximum MAP@1000 for high values of α, the performance of LSE, on the other hand, reaches a optimum for relatively low values of α (≈ 0.3) and then degrades quickly to a MAP@1000 level below the effectiveness of text-only ranking (i.e., $\alpha = 0.0$). Interestingly enough, NVSM does not exhibit this behavior as it seems to converge back to the text-only performance as α approaches 1. If we look at the different benchmarks, we observe similar behavior except for the smallest domain (Pet Supplies). Surprisingly, in the Pet Supplies department, we observe that MAP@1000 remains stable for values of α close to 0.9 for both LSE and NVSM. This is likely due to a very specific characteristic of pet supplies. That is, the department is itself divided in mostly mutually-exclusive sub-departments based on animal species and the product substitution relations are very likely to encode this disjoint property. Fig. 2 provides a visualization of the product representations that shows the formation of sub-classes according to the animal taxonomy in the Pet Supplies department. In particular, we see that—while there is some overlap between cat and dog supply products—there are exclusive clusters for bird, fish, horse and small animal supplies. Consequently, queries that semantically correspond to a particular animal species are likely to be directly projected into the cluster corresponding to that animal. The stable increase in retrieval effectiveness visible in Fig. 1a is obtained as

[5] https://github.com/usnistgov/trec_eval

Table 2: Comparison of extrinsic methods (g_{struct} and g_{match}) to incorporate entity similarity within product search for various retrieval models (§4.3.1) and the intrinsic incorporation (Eq. 6) of entity similarity within neural vector space models (LSE, NVSM). The framework we present in this paper—where entity similarity is incorporated intrinsically—is not compatible with the BM25, QLM and w2v retrieval models. Performance is reported on the test set, for the hyperparameter configuration (§4.3) optimized on the validation set (§4.2.1) in terms of *MAP*@1000. Significance (§4.4) is computed between extrinsic (g_{match}) and intrinsic incorporation of entity similarity for NVSM.

	Pet Supplies			Sports & Outdoors			Toys & Games			Electronics		
	MAP	NDCG	P@10	MAP	NDCG	P@10	MAP	NDCG	P@10	MAP	NDCG	P@10
BM25												
Text-only	0.139	0.270	0.219	0.119	0.237	0.165	0.089	0.201	0.181	0.070	0.186	0.201
Extrinsic (g_{struct})	0.124	0.258	0.200	0.118	0.235	0.163	0.084	0.192	0.153	0.067	0.180	0.179
Extrinsic (g_{match})	0.139	0.275	0.220	0.122	0.241	0.166	0.091	0.203	0.182	0.076	0.191	0.203
QLM (d)												
Text-only	0.131	0.260	0.212	0.106	0.225	0.153	0.080	0.190	0.181	0.069	0.187	0.202
Extrinsic (g_{struct})	0.114	0.237	0.169	0.100	0.217	0.143	0.076	0.176	0.138	0.067	0.179	0.172
Extrinsic (g_{match})	0.135	0.263	0.209	0.109	0.227	0.154	0.083	0.186	0.184	0.076	0.194	0.201
w2v (add)												
Text-only	0.134	0.246	0.186	0.098	0.198	0.134	0.080	0.183	0.170	0.065	0.167	0.187
Extrinsic (g_{struct})	0.126	0.236	0.164	0.096	0.197	0.132	0.079	0.180	0.164	0.064	0.166	0.179
Extrinsic (g_{match})	0.131	0.241	0.184	0.098	0.198	0.134	0.081	0.184	0.170	0.067	0.171	0.189
w2v (si)												
Text-only	0.141	0.258	0.191	0.122	0.233	0.155	0.107	0.221	0.207	0.085	0.196	0.203
Extrinsic (g_{struct})	0.136	0.251	0.180	0.121	0.231	0.151	0.104	0.215	0.194	0.084	0.194	0.197
Extrinsic (g_{match})	0.142	0.258	0.191	0.122	0.233	0.155	0.108	0.220	0.207	0.086	0.197	0.203
LSE												
Text-only	0.128	0.249	0.201	0.097	0.197	0.137	0.080	0.180	0.165	0.059	0.160	0.172
Extrinsic (g_{struct})	0.122	0.240	0.188	0.096	0.195	0.135	0.077	0.170	0.143	0.058	0.148	0.126
Extrinsic (g_{match})	0.129	0.249	0.201	0.097	0.197	0.137	0.081	0.180	0.165	0.062	0.161	0.170
Intrinsic	0.134	0.259	0.206	0.082	0.165	0.109	0.077	0.173	0.172	0.069	0.162	0.168
NVSM												
Text-only	0.205	0.344	0.272	0.176	0.308	0.211	0.145	0.267	0.232	0.105	0.225	0.235
Extrinsic (g_{struct})	0.197	0.333	0.253	0.174	0.306	0.207	0.140	0.257	0.205	0.103	0.220	0.217
Extrinsic (g_{match})	0.205	0.343	0.272	0.176	0.308	0.211	0.145	0.267	0.232	0.107	0.226	0.235
Intrinsic	**0.233*****	**0.379*****	**0.307*****	**0.201*****	**0.331*****	**0.223*****	**0.155****	**0.274**	**0.239**	**0.117*****	**0.234****	**0.249****



(a) Pet Supplies **(b) Sports & Outdoors** **(c) Toys & Games** **(d) Electronics**

Figure 1: The effect of the trade-off parameter α between text matching and product similarity signals (§6) within latent vector space models. Retrieval effectiveness is reported in terms of MAP@1000 on the test set. For every value of trade-off parameter α, we choose the remaining hyperparameters (§4.3) based on the validation set (§4.2.1) in terms of MAP@1000 (§4.2.2).

mixing the product substitutability relations with the text objective further encourages the clustering behavior that follows the animal classification system. This is because product substitution relations in the Pet Supplies department typically relate only products that are applicable to a particular animal species (i.e., supplies that apply to multiple species are rare). That is, within the Pet Supplies benchmark used in this paper (Table 1), 82.47% of all relations are between pet supply products meant for the same species.

Following these observations, we make the following recommendations. First of all, we confirm that NVSM is a superior neural vector space model over LSE [53]. Secondly, when using NVSM, a non-optimal choice of trade-off parameter α does not seem to considerably impact retrieval effectiveness negatively (Fig. 1). Consequently, we recommend setting the trade-off parameter α as 0.8. This recommendation, however, only holds if the incorporated similarity relations correlate with relevance, as is the case with substitutable products. One question that remains is how the intrinsic incorporation of product similarity influences the product rankings shown to the users. We address this question in the next section.

5.3 Effect on product rankings

RQ3 Table 3 shows the difference in perplexity of language models (§4.2.2) estimated on rankings generated by latent vector space models estimated using only the text objective (α = 0.0) and using a mixture of text and similarity objectives (α optimized on validation set).

We observe that rankings generated by similarity-incorporating vector space models have less uncertainty in their language models

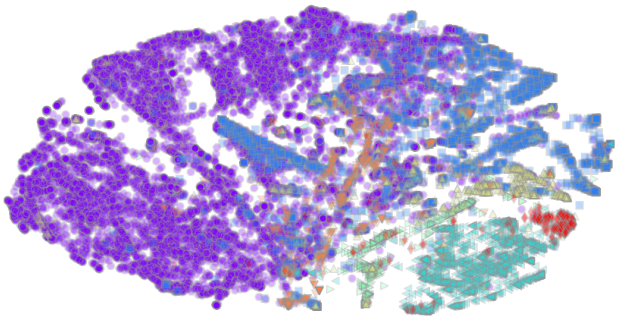


Figure 2: Visualization of learned product representation in the Pet Supplies department using t-SNE [27]. Product representations cluster by animal taxonomy: dogs (purple circle), cats (blue square), fish (cyan left-arrow), birds (green right-arrow), small animals (yellow up-arrow), reptiles & amphibians (orange down-arrow) and horses (red diamond). While there is some overlap between cat and dog supply products, we can see that there are exclusive clusters for bird, fish, horse and small animal supplies. Best viewed in color.

Table 3: Average differences in perplexity on a categorical language model estimated on the top-100 documents returned by latent vector space models with and without the intrinsic incorporation of product similarity. Note that here we report only the effect size and not the perplexities of the individual systems. Significance (§4.4) is computed between the rankings with and without the intrinsic incorporation of product similarity.

	LSE	NVSM
Pet Supplies	−0.149	-4.162^{***}
Sports & Outdoors	-1.734^{***}	-2.147^{***}
Toys & Games	−0.691	-1.717^{***}
Electronics	-5.554^{***}	+0.181

than rankings generated by vector space models estimated on text only. For NVSM, this trend is significant ($p < 0.05$) for three out of four benchmarks. For the largest benchmark, Sports & Outdoors, we observe a slight increase in language model perplexity when using NVSM. A significant difference, however, was not observed in this case ($p \geq 0.05$). As the language of product rankings becomes easier to predict (i.e., less entropy) when incorporating product substitutability, we conclude that the incorporation of a notion of product similarity that encodes substitutability generates less diverse rankings. In effect, this finding relates back to the trade-off between relevance and diversity discussed in the literature [8]. This finding is not surprising. In fact, it follows directly from the definition of product substitutability that we discussed in the introduction, namely: "two goods (i.e., products) are substitutes if both can satisfy the same need to the consumer" [23]. Consequently, within product search, information and consumption needs are two sides of the same coin.

6 CONCLUSIONS

We introduced a framework for incorporating product substitutability within latent vector space models. In particular, we investigated whether the incorporation of product entity substitutability can improve relevance ranking effectiveness within product search. We also investigated the trade-off between text matching and entity similarity objectives and provide insights within the product search domain. Finally, we performed an analysis of the effect of incorporating product substitutability on product search rankings.

We found that the incorporation of product substitutability relations within latent vector space models improves the relevance of product rankings w.r.t. to a user-issued query, albeit at the cost of product diversity. The trade-off between text matching and entity similarity objectives has a different effect depending on the text matching objective that is used. For LSE, an optimal MAP@1000 is achieved when more weight is put on the text objective, whereas for NVSM, more reliance on the similarity objective yields better rankings. In fact, NVSM shows stable retrieval effectiveness when more reliance is put on the similarity objective and performance rarely degrades under the performance level that was achieved when estimating NVSMs with text only (Fig. 1). For one particular product department, Pet Supplies, we found that retrieval effectiveness kept improving when more reliance was put on entity similarity. We found that this is because the domain is naturally divided in mutually-exclusive sub-departments that correspond to the taxonomy of animals. Consequently, incorporating product substitutability caused the creation of tightly-connected clusters that correspond to animal species. Furthermore, product rankings obtained by querying latent vector space models with incorporated product substitutability relations use more predictable language than vector space models estimated without the similarity relations. We conclude that improvements in search relevance come at the cost of ranking diversity. This observation is not unexpected, as it is the exact embodiment of product substitutability. More precisely, "two goods (i.e., products) are substitutes if both can satisfy the same need to the consumer" [23] and, within product search, information and consumption needs are two sides of the same coin.

The broader implications of our work are three-fold. First, we verified the substitutability principle from economics within e-commerce search and confirm the findings of McAuley et al. [30] who suggest that e-commerce product substitutability can be inferred from user interactions. Second, we showed that the incorporation of similarity within latent retrieval spaces can improve retrieval effectiveness. This raises further questions regarding what types of similarity can yield improvements for which retrieval scenarios and how the semantics of these similarity relations tie back into the particular retrieval domain. For example, similarity between entities can be computed from non-textual data (e.g., images) and then be used to incorporate image similarity within the retrieval stage. Finally, on the topic of latent vector space models, Fig. 2 reveals more than meets the eye. Current latent vector space models work well when user queries correspond to a single coherent topic. This comes as little surprise, as every query is projected into a single point within the retrieval space that is surrounded by documents of a particular topic. Consequently, one can wonder whether projecting a single query to multiple points in the retrieval space can further improve retrieval effectiveness.

Our work is limited by the fact that we only considered product substitutability relations. In the case of product search, product complementarity might provide a good source to improve ranking diversity. The question remains whether similarity can be incorporated for ad-hoc document retrieval and how this similarity should be obtained. The cluster hypothesis [54] might be of use

here. Another limitation of our work over extrinsic methods is that, as product substitutability relations are incorporated during model estimation, models need to be re-trained from scratch as more similarity information becomes available.

Future work includes the investigation of (a) *complementary product relations* and in what way they influence ranking relevance and diversity. Unfortunately, the Amazon product data lacks product diversity ground-truth, (b) *pseudo similarity feedback* where latent vector space models are iteratively trained and the similarity between entities/documents within the current iteration is constructed using their vector space similarity of the previous iteration, (c) the effect of *incorporating lexical vector space similarity* into latent vector space models for ad-hoc document retrieval, and (d) *multiple projections of queries* into latent retrieval spaces so as to capture the multi-topicality of queries.

Acknowledgements

We would like to thank Manos Tsagkias and the anonymous reviewers for their valuable comments. This research was partially supported by the Bloomberg Research Grant program, and the Google Faculty Research Awards program. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Q. Ai, L. Yang, J. Guo, and W. B. Croft. Analysis of the paragraph vector model for information retrieval. In *ICTIR*, pages 133–142. ACM, 2016.
[2] Q. Ai, L. Yang, J. Guo, and W. B. Croft. Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *SIGIR*, pages 869–872. ACM, 2016.
[3] Q. Ai, Y. Zhang, K. Bi, X. Chen, and B. W. Croft. Learning a hierarchical embedding model for personalized product search. In *SIGIR*, 2017.
[4] P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379, 2010.
[5] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. Techn. report, W3C, http://www.w3.org/TR/owl-ref/, Feb. 2004.
[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3: 993–1022, 2003.
[7] M. Bron, K. Balog, and M. de Rijke. Ranking related entities: Components and analyses. In *CIKM*, pages 1079–1088. ACM, October 2010.
[8] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336. ACM, 1998.
[9] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
[10] 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.
[11] H. Duan and C. Zhai. Mining coordinated intent representation for entity search and recommendation. In *CIKM*, pages 333–342. ACM, 2015.
[12] 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.
[13] 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.
[14] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 22:457–479, 2004.
[15] R. H. Frank. *Microeconomics and behavior (9th edition)*. McGraw-Hill Irwin, 2010.
[16] J. Henderson and R. Quandt. *Microeconomic theory: a mathematical approach*. Economics handbook series. McGraw-Hill, 1980.
[17] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.
[18] 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.
[19] T. Kenter and M. de Rijke. Short text similarity with word embeddings. In *CIKM*, pages 1411–1420. ACM, 2015.
[20] O. Kurland. The cluster hypothesis in information retrieval. In *ECIR*, pages 823–826, 2014.
[21] O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *SIGIR*, pages 194–201. ACM, 2004.
[22] O. Kurland and L. Lee. Pagerank without hyperlinks: Structural reranking using links induced by language models. *TOIS*, 28(4):18:1–18:38, Nov. 2010.
[23] J. M. Lattin and L. McAlister. Using a variety-seeking model to identify substitute and complementary relationships among competing products. *Journal of Marketing Research*, pages 330–339, 1985.
[24] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.
[25] H. Li and J. Xu. Semantic matching in search. *Found. & Tr. in Information Retrieval*, 7(5):343–469, June 2014.
[26] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.
[27] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov): 2579–2605, 2008.
[28] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.
[29] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *KDD*, pages 785–794. ACM, 2015.
[30] 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.
[31] S. McPartlin, L. F. Dugal, M. Jenson, and I. W. Kahn. Understanding how US online shoppers are reshaping the retail experience. Pricewaterhouse Coopers, 2012.
[32] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient estimation of word representations in vector space. arXiv 1301.3781, 2013.
[33] G. Montavon, G. B. Orr, and K.-R. Müller. *Neural Networks: Tricks of the Trade*. Springer, 2012.
[34] P. Nurmi, E. Lagerspetz, W. Buntine, P. Floréen, and J. Kukkonen. Product retrieval for grocery stores. In *SIGIR*, pages 781–782. ACM, 2008.
[35] U. D. of Agriculture. Commodity and food elasticities database, 2017. Accessed 26-July-2017.
[36] H. Raviv, O. Kurland, and D. Carmel. The cluster hypothesis for entity oriented search. In *SIGIR*, pages 841–844. ACM, 2013.
[37] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Found. & Tr. in Inform. Retr.*, 3(4):333–389, 2009.
[38] J. Rowley. Product search in e-shopping: a review and research propositions. *Journal of Consumer Marketing*, 17(1):20–35, 2000.
[39] R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approximate Reasoning*, 50(7):969–978, 2009.
[40] S. K. K. Santu, P. Sondhi, and C. Zhai. On application of learning to rank for e-commerce search. In *SIGIR*, Aug. 2017.
[41] 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.
[42] C. G. Snoek, M. Worring, and A. W. Smeulders. Early versus late fusion in semantic video analysis. In *MM*, pages 399–402. ACM, 2005.
[43] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *ICIA*, 2005.
[44] M. Szummer and E. Yilmaz. Semi-supervised learning to rank with preference regularization. In *CIKM*. ACM, 2011.
[45] A. Tonon, G. Demartini, and P. Cudré-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *SIGIR*, pages 125–134. ACM, 2012.
[46] A. Trotman, J. Degenhardt, and S. Kallumadi. The architecture of ebay search. In *SIGIR 2017 Workshop on eCommerce (eCom'17)*, Aug. 2017.
[47] U.S. Department of Commerce. Quarterly Retail E-Commerce Sales. https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf, 2017. Accessed 26-July-2017.
[48] C. Van Gysel, M. de Rijke, and E. Kanoulas. Learning latent vector spaces for product search. In *CIKM*, pages 165–174. ACM, 2016.
[49] C. Van Gysel, M. de Rijke, and M. Worring. Unsupervised, efficient and semantic expertise retrieval. In *WWW*, pages 1069–1079. ACM, 2016.
[50] C. Van Gysel, M. de Rijke, and E. Kanoulas. Structural regularities in text-based entity vector spaces. In *ICTIR*, pages 3–10. ACM, 2017.
[51] C. Van Gysel, M. de Rijke, and E. Kanoulas. Pyndri: a python interface to the indri search engine. In *ECIR*. Springer, 2017.
[52] C. Van Gysel, M. de Rijke, and E. Kanoulas. Semantic entity retrieval toolkit. In *Neu-IR 2017*, 2017.
[53] C. Van Gysel, M. de Rijke, and E. Kanoulas. Neural vector spaces for unsupervised information retrieval. *TOIS*, 36(4):Article 38, 2018.
[54] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.
[55] D. Vandic, F. Frasincar, and U. Kaymak. Facet selection algorithms for web product search. In *CIKM*, pages 2327–2332. ACM, 2013.
[56] I. Vulić and M.-F. Moens. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *SIGIR*, pages 363–372. ACM, 2015.
[57] Wikipedia. Wikipedia:redirect — wikipedia, the free encyclopedia, 2017. Accessed 10-July-2017.
[58] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *TOIS*, 22(2):179–214, 2004.