MEmbER: Max-Margin Based Embeddings for Entity Retrieval

Shoaib Jameel, Zied Bouraoui, Steven Schockaert
School of Computer Science and Informatics
Cardiff University
Queen's Buildings, 5 The Parade, Roath
Cardiff, United Kingdom. CF24 3AA
{JameelS1,bouraouiz,SchockaertS1}@cardiff.ac.uk

ABSTRACT

We propose a new class of methods for learning vector space embeddings of entities. While most existing methods focus on modelling similarity, our primary aim is to learn embeddings that are interpretable, in the sense that query terms have a direct geometric representation in the vector space. Intuitively, we want all entities that have some property (i.e. for which a given term is relevant) to be located in some well-defined region of the space. This is achieved by imposing max-margin constraints that are derived from a bag-of-words representation of the entities. The resulting vector spaces provide us with a natural vehicle for identifying entities that have a given property (or ranking them according to how much they have the property), and conversely, to describe what a given set of entities have in common. As we show in our experiments, our models lead to a substantially better performance in a range of entity-oriented search tasks, such as list completion and entity ranking.

CCS CONCEPTS

•**Information systems** → **Information retrieval;** *Web searching and information discovery;* •**Computing methodologies** → Natural language processing;

KEYWORDS

Entity embedding, maximum margin, list completion, entity ranking

ACM Reference format:
Shoaib Jameel, Zied Bouraoui, Steven Schockaert. 2017. MEmbER: Max-Margin Based Embeddings for Entity Retrieval. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080803

1 INTRODUCTION

Entity retrieval (ER) aims at improving traditional document retrieval by directly presenting the user with a list of relevant entities. Existing work in this domain has largely focused on two classes of methods, which differ in the kind of information that is utilized. Methods in the first class, which we will refer to as *document-centric ER*, mostly rely on textual descriptions. For example, we can identify relevant entities by matching their Wikipedia article with the query terms [11, 13, 15], or we can identify experts on a given topic by matching their homepage with the topic description [1]. Document-centric ER methods thus remain close to document retrieval models, but typically augment these models with some kind of context information (e.g. a bag-of-words representation of the Wikipedia categories to which an entity belongs, or the department where a potential expert is working). Methods in the second class, which we will refer to as *structured ER*, rely on structured knowledge that is available about the entities [6, 32, 40], e.g. in knowledge graphs such as DBpedia, Freebase, or WikiData, as semantic markup (e.g. RDFa), or from applying information extraction methods.

Clearly, structured ER methods can only identify relevant entities if knowledge is available about the criteria expressed in the query. For example, the query "mountains in Japan" can be answered correctly by using the fact that entities such as *Mount Fuji* have value *Japan* for the property *country* in WikiData[1]. However, the similar query "mountains in South Japan" cannot be interpreted, as no information is given about where in the country each mountain is located. Document-centric ER methods are less constrained by the types of queries they can interpret, but they still rely on query terms being explicitly mentioned in the available textual descriptions of the relevant entities. While this is a typical problem for information retrieval systems, it is particularly acute in this context, when we want to include criteria that are subjective, vague, or otherwise a matter of degree. For example, queries such as "tall mountains in Japan" or "influential music bands" are problematic for current approaches, as e.g. Wikipedia articles would rarely explicitly state that a mountain is tall or that a band is influential. Note that hybrid models, which combine structured information with bag-of-words representations, cannot solve this issue since such criteria are missing from structured knowledge bases as well.

The solution we propose is to complement the structured knowledge and bag-of-words representations with a low-dimensional vector space embedding. As the purpose of this embedding is specifically to answer queries, we need a mechanism to map query terms to geometric objects in the vector space. The idea is illustrated in Figure 1, which depicts an embedding of tourist attractions. In particular, tourist attractions are represented as points in this space, while relative properties such as 'old' or 'famous' are represented as vectors. These vectors allow us to rank the entities according to how much they have the corresponding property (e.g. according

SIGIR '17, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080803

[1] https://www.wikidata.org/wiki/Q39231

to this embedding, *eiffel tower* is more famous than *stonehenge*, which is more famous than *notre dame de paris*). In addition, there are regions in this space which group the entities that share some well-defined property (e.g. being a religious site). We will call such a representation, in which the salient properties of the domain are explicitly modelled as vectors and regions, an *interpretable entity embedding*. Clearly, interpretable entity embeddings offer a natural way of answering queries such as "famous tourist sites in France". Our main focus in this paper will be on how such embeddings can be learned.

It was proposed in [16] that vectors corresponding to salient properties can be discovered, in a given entity embedding, by finding a hyperplane which separates the entities that have a given term in their bag-of-words representation from the entities that do not. Since we initially do not know which terms describe salient properties, a separating hyperplane is obtained for each (sufficiently frequent) term in the collection. Then, those terms whose associated hyperplane is sufficiently successful in separating the two groups of entities are assumed to correspond to salient properties (modelled by the normals of the hyperplanes). The underlying intuition is that while e.g. most Wikipedia articles may not explicitly mention that a mountain is tall, the term *tall* will still be mentioned in some articles, especially in cases where the tallness is what sets the mountain apart from other mountains. As a result, the normal of the separating hyperplane for the term *tall* will be directed towards the region of the space where these exceptionally tall mountains occur. While promising results were reported in [16], an important limitation of the method is that it has to rely on existing entity embedding methods, which are aimed at modelling similarity. It is unclear whether such entity embeddings are optimal in an ER context, however, where the focus is not on measuring similarity, but on ranking entities and on identifying entities that have a given property.

The main research question we study in this paper is whether we can obtain better results by learning interpretable entity embeddings in a more direct way. In particular, we introduce a novel model for learning entity embeddings, which uses max-margin constraints to encode the desideratum that (salient) properties of entities should have a simple geometric representation in the entity embedding. In the most basic variant, we associate with each term a separating hyperplane, in the spirit of [16], but with the difference that the entity representation and separating hyperplanes are jointly learned. Given the central role of ranking in ER, we also consider a variant that associates with each term a sequence of parallel hyperplanes, partitioning the space based on how strongly entities are associated with the term. Finally, we also consider the use of quadratic kernels, which allows us to go beyond hyperplanes, and also represent terms using hyper-ellipsoids and other quadratic hypersurfaces.

The remainder of this paper is structured as follows. In Section 2 we discuss related work on vector space embeddings and entity retrieval. We then explain the details of our model in Section 3, after which we present our experimental results in Section 4.

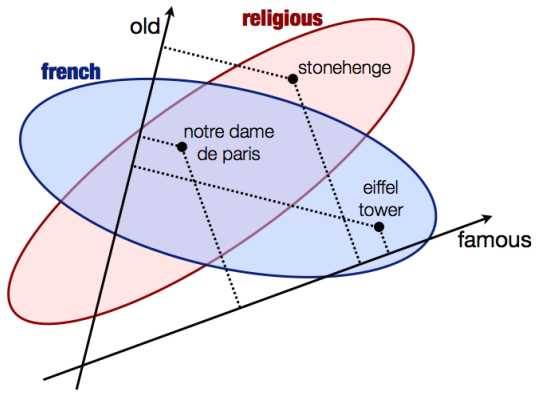

Figure 1: Illustration of an interpretable entity embedding.

2 RELATED WORK

2.1 Entity retrieval

Entity retrieval refers to a broad class of information retrieval methods in which entities play a central role. Within this class, as already mentioned, we can distinguish between structured and document-centric approaches.

Structured ER approaches typically address the problem of identifying entities from a knowledge graph (i.e. a set of subject-predicate-object triples) that are relevant to a given keyword query. A common strategy consists of first representing the relevant information about each entity as a structured document. Initial results can then be obtained by relying on ranking functions for (structured) document retrieval, such as BM25F, which can subsequently be refined using the structured information that is available about the entities [6, 32, 34, 37, 40, 48].

Several authors have used some kind of similarity-based model to improve the results from some baseline model. For example, in [47], given an initial set of candidate entities, a subgraph of the overall knowledge graph is determined, which is then represented in a low-dimensional vector space using RESCAL [33]. This vector space is used to score entities according to how close they are to the top-3 entities retrieved using the baseline model. Finally, this score is combined with a standard document retrieval model using a learning-to-rank approach. In this paper, we will use a similar strategy for demonstrating the potential of our entity embeddings for entity ranking tasks. However, as we will see, pure knowledge graph embedding methods, such as RESCAL, are actually not optimal. Instead, where possible, better results can be obtained by learning vector space representations from a combination of structured information and textual descriptions. As an alternative to using vector space embeddings, in [19] entities are clustered (in an offline preprocessing step) based on a combination of structured information and textual descriptions, and these clusters are then similarly used to expand the results returned by a baseline model. Along similar lines, the approach proposed in [10] uses predetermined groups of semantically related entities as part of the retrieval model.

Document-centric ER approaches also respond to queries by providing a ranked list of entities, but they primarily rely on textual

descriptions (e.g. a Wikipedia article). This is the setting that was considered, among others, in the INEX entity ranking track [11, 13, 15]. While document-centric ER could be treated as a standard document retrieval task, using e.g. Wikipedia as the collection, most approaches make use of some structured information that is available about the entities, such as Wikipedia categories or information derived from the Wikipedia link structure [2, 14, 24].

A difference is sometimes made between 'Wikipedia entity ranking' (i.e. identifying relevant entities based on Wikipedia articles) and 'web entity ranking' [14]. In the latter case, which is the scenario that was considered in the TREC entity ranking track [3, 4], the aim is to find a list of relevant entities in a general web collection and to represent each entity by its most authoritative homepage. This task is harder than Wikipedia entity ranking, as it involves additional challenges such as deciding whether a given website is a homepage representing an entity, and if so, what is the semantic type of that entity. However, it was shown in [14] that web entity ranking can often be solved effectively by first finding a relevant Wikipedia page, and then using that Wikipedia page to find a more authoritative homepage, if one exists. This illustrates the unique central role of Wikipedia for entity retrieval. It should be noted, however, that Wikipedia based retrieval will only be effective for entities that are of general interest. In domain-specific ER tasks such as expert finding [1], for instance, Wikipedia-centric approaches are obviously not suitable.

A slightly different task is considered in [38], which considers the problem of identifying all entities that are relevant to a given query. In other words, while the aforementioned works are concerned with retrieving entities of a particular semantic type, which serve as answers to the query, the task proposed in [38] is to identify those entities in the top-ranked documents that are salient to the query. This involves named entity recognition and entity linking (i.e. assigning the named entity to its Wikipedia article), and then filtering the remaining entities in some way. A similar task was considered in [17], where the focus is rather on determining the saliency of the entities that are mentioned in a given news article.

Finally, several approaches to query expansion have been proposed that are based on linking entities occurring in the query to Wikipedia or to structured knowledge bases [28, 31, 36, 44]. While such methods are not concerned with entity retrieval, the success of these methods suggests that entity embeddings may also have an important role to play for ad hoc document retrieval [29].

2.2 Vector space embeddings

Our approach is related to two popular types of vector space representations: word embeddings and knowledge graph embeddings.

Word embeddings are low-dimensional vector space representations of words, which are learned such that similar words are represented by similar vectors. One of the most popular methods to obtain such representations is the Skip-gram (SG) model, as well as the related Continuous Bag Of Words (CBOW) model, which are often jointly referred to as word2vec [30]. These models have evolved from (computationally more expensive) neural network based language models [5]. Another possible strategy is to factorize word-word co-occurrence matrices [41]. Typically, singular value decomposition is used as the factorization method and co-occurrence statistics are expressed using Positive Pointwise Positive Information (PPMI). Interestingly, it has been shown that SG can be expressed as a matrix factorization model based on a slight variation of PPMI [26].

Despite the focus on similarity, it has been observed that the word embeddings produced by methods such as SG capture several other kinds of linguistic regularity [30]. Most notably, analogous word pairs tend to correspond to approximately parallel vectors, e.g. writing v_w for the vector representation of a word v, it is found that $v_{paris} - v_{france} \approx v_{rome} - v_{italy}$. This effect has been analysed, among others, in [35], where a word embedding model called GloVe was introduced, aimed specifically at learning such linear regularities.

Several authors have already studied the use of word embeddings in the context of information retrieval, e.g. for query expansion [45], to model dependencies between terms in language models [20], or for weighting query terms [46].

Knowledge graph embeddings encode a given set of subject-predicate-object triples in a low-dimensional vector space. In this way, statistical regularities that are implicit in the data can be made explicit, allowing us to obtain triples which are missing from the knowledge graph but are nonetheless plausible (e.g. knowing that person x works for company y and that y is based in city z, we may plausibly derive that x lives in z), or identify triples from the knowledge graph which are likely to be wrong. One possibility to learn a knowledge graph embedding, which was adopted in the RESCAL model [33], is to represent the triples as a tensor and rely on tensor factorization to reduce the dimensionality, similar to how singular value decomposition is used to obtain low-dimensional document representations in Latent Semantic Analysis [12].

In recent years, better results have been obtained by learning the embedding in a more explicit way. For example, the popular TransE model [7] associates with each entity e a point p_e and with each predicate r a vector v_r, such that $p_f \approx p_e + v_r$ iff the triple (e, r, f) is asserted, meaning that entities e and f stand in relation r. Despite its conceptual simplicity, this model was found to be surprisingly effective. A crucial limitation, however, is that it can only faithfully capture one-to-one relations. To address this, a large number of approaches have been proposed in recent years, including TransH [43], which projects entities on a hyperplane before applying the translation, and TransR [27], which more generally applies a relation-specific linear transformation before applying the translation.

Note that knowledge graph embeddings only capture structured information about entities. In the context of ER, however, a lot of relevant information is usually only expressed in a textual form. It has been shown that such textual descriptions can be used to improve the quality of entity embeddings. For example, in [42] and [49] a model is proposed which combines a component that is similar in spirit to Skip-gram, to exploit textual descriptions of entities, with a component that is similar in spirit to TransE, to exploit the triples that are available. It is shown that incorporating textual descriptions indeed improves the predictive performance of the embedding. In [23] similarly a word embedding component (based on GloVe) is combined with a component that captures structured information about the entities. As the latter model has an explicit representation of semantic types, we will use it in the next section as the starting point for learning interpretable embeddings.

3 ENTITY EMBEDDING MODELS

In this section, we first review EECS, the entity embedding method that was proposed in [23]. Subsequently, we show how these limitations can be addressed in a natural way by using max-margin constraints to directly encode the idea that terms should have a clear geometric representation in the vector space. We also consider a variant which is inspired by ordinal regression SVM models [39], and provide a detailed comparison with the EECS model.

3.1 The EECS model

Let $\mathcal{E}$ be a set of entities, $\mathcal{S}$ a set of semantic types and $\mathcal{R}$ a set of relations. The aim of the model proposed in [23], called EECS, is to represent each entity e from $\mathcal{E}$ as a point $p_e \in \mathbb{R}^n$, such that all entities of the same semantic type belong to some lower-dimensional subspace. The aim of their work is to assess the suitability of these subspaces as approximations of conceptual spaces [21]. The latter are a kind of geometric representations which are used to model cognitive phenomena such as induction, vagueness, typicality and metaphors. Conceptual spaces closely correspond to what we referred to as interpretable entity embeddings in the introduction, in the sense that entities are represented as points, and properties and categories are represented as regions. Note, however, that conceptual spaces model entities of one particular semantic type only, unlike the entity embeddings that are used in natural language processing and information retrieval. The EECS model attempts to provide a bridge between conceptual spaces and vector space models by viewing conceptual spaces as being themselves embedded in a more general vector space.

The input to EECS consists of a bag-of-words representation W_e of every entity e, a set of entities $\mathcal{E}_s$ for each semantic type $s \in \mathcal{S}$ and a set of knowledge graph triples $\mathcal{T}$ of the form (e, r, f), with $e, f \in \mathcal{E}$ and $r \in \mathcal{R}$. The vector space representation is obtained by minimizing an objective function of the following form

$$J = \alpha(J_{text} + J_{glove}) + (1-\alpha)(J_{type} + J_{rel}) + \beta J_{reg} \quad (1)$$

with $\alpha \in [0,1]$ and $\beta \geq 0$. The component J_{text} intuitively imposes the view that entities with similar bag-of-words representations should be represented using similar vectors. The terms from these bag-of-words representations are represented using the GloVe model, as expressed by the component J_{glove}:

$$J_{glove} = \sum_{t_i \in W} \sum_{t_j \in W} f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + b_j - \log x_{ij})^2$$

where W represents the vocabulary. In the GloVe model, each term t_i is represented using two vectors w_i and $\tilde{w}_j$, where w_i intuitively represents the meaning of t_i and $\tilde{w}_j$ represents how the occurrence of t_i in the context of another word affects the meaning of that other word. Furthermore, b_i and b_j represent (scalar) bias terms, while x_{ij} is the number of times t_i occurs in the local context of t_j in the corpus. Finally, $f(x_{ij})$ is a weighting function aimed at limiting the impact of pairs (t_i, t_j) for which x_{ij} is small:

$$f(x_{ij}) = \begin{cases} \left(\frac{x_{ij}}{x_{max}}\right)^{\alpha} & \text{if } x_{ij} < x_{max} \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

The values x_{max} and α are tuned, with $x_{max} = 100$ and $\alpha = 0.75$ often giving good results. Note in particular that pairs for which $x_{ij} = 0$ are thus omitted entirely. The component J_{text} parallels the GloVe model, replacing the target word t_i with an entity:

$$J_{text} = \sum_{e_i \in \mathcal{E}} \sum_{t_j \in W_{e_i}} f(y_{ji})(p_{e_i} \cdot \tilde{w}_j + b_i + b_j - \log y_{ji})^2$$

where p_{e_i} is the representation of entity e_i (viewed as a vector) and y_{ji} is the number of times t_j appears in the bag-of-words representation W_{e_i}.

The component J_{type} simply expresses that for each semantic type s there are points $q_1^s, ..., q_n^s$ such that for every $e \in \mathcal{E}_s$ it holds that p_e is in the convex hull of $q_1^s, ..., q_n^s$. By itself this constraint is trivial. However, the component J_{reg} additionally expresses that the space spanned by $q_1^s, ..., q_n^s$ is as low-dimensional as possible, using nuclear norm regularization [18].

Finally, the component J_{rel} is similar in spirit to the TransE model. Specifically, it expresses that for each relation r we can find a vector v_r such that all entities in $\mathcal{E}_{(e,r,.)} = \{f \mid (e,r,f) \in \mathcal{T}\}$, for a given e, are close to $e + v_r$ and, similarly, that all entities in $\mathcal{E}_{(.,r,f)} = \{e \mid (e,r,f) \in \mathcal{T}\}$, for a given f, are close to $f - v_r$. In addition, similarly to the component J_{type} it expresses that the entities in $\mathcal{E}_{(e,r,.)}$ and the entities in $\mathcal{E}_{(.,r,f)}$ should both belong to some lower-dimensional subspace. We refer to [23] for more details on the components J_{type}, J_{rel} and J_{reg}, as we will not be concerned with how semantic types and knowledge graph triples are modelled.

3.2 Max-margin based embeddings

Our aim is to learn entity embeddings in which terms, and by extension properties of entities, have a natural geometric interpretation. Central to this view is the requirement that entities to which a given term applies can be separated from entities to which the term does not apply. If we make the simplifying assuming that a term applies to an entity e iff it occurs at least once in the bag-of-words representation W_e, this requirement can be encoded using a squared hinge loss function as follows:

$$J_{text}^{class} = \sum_{t \in W} \frac{\sigma(t)}{|\mathcal{E}_t|} \sum_{e \in E_t} \max\left(0, 1 - I_e^t(K(p_e, \tilde{w}_t) + b_t)\right)^2 + \lambda \|\tilde{w}_t\|^2$$

where $I_e^t = 1$ if $t \in W_e$ and $I_e^t = -1$ otherwise. As before, we write p_e for the representation in the vector space of entity e, $\tilde{w}_t$ is a vector reflecting what information can be derived about entity e from the fact that term t occurs in W_e, and b_t is a bias term. We will assume that the kernel K in the expression above is either linear or quadratic. These kernels have the benefit that the vector $\tilde{w}_t$ has a clear geometric interpretation in the entity embedding, defining a hyperplane in the case of a linear kernel and a convex region (with a quadratic hypersurface) in the case of a quadratic kernel. Finally, σ is a weighting function.

Intuitively, we can think of $\mathcal{E}_t$ as the set of all entities. However, with this choice, optimizing the expression J_{text}^{class} would be too time-consuming, as it would involve a constraint for each term-entity pair. Therefore in $\mathcal{E}_t$ we only include (i) those entities e for which $t \in W_e$ and (ii) a random sample of entities for which $t \notin W_e$. This is similar in spirit to the use of negative sampling in the Skip-gram model, and in fact, also avoids problems with class imbalance. In our experiments, we choose twice as many negative examples as there are positive examples.

Note that in J_{text}^{class} both the representations of the entities p_e and the hyperplanes (defined by $\tilde{w}_t$ and b_t) are unknowns. This stands in contrast to most applications of max-margin constraints, where the entity representations are usually given and only the separating hyperplanes need to be found. By jointly learning the entity representations and hyperplanes, we can obtain a space which is directly optimized to represent terms as geometric objects. Furthermore, note that by using the squared hinge loss, the component J_{text}^{class} is compatible with the other components of the model, in particular J_{type} and J_{rel}, in the sense that all components of the model can then be interpreted in terms of the squared distance between entity representations and geometric objects (e.g. subspaces in the case of J_{type} and hyperplanes in the case of J_{text}^{class}). Furthermore, since the squared hinge loss is differentiable, this also avoids the need for subgradient-based optimization methods.

The use of a weighting function σ allows us, in principle, to focus the model on the most informative terms. In practice, however, we typically have no prior knowledge about the informativeness of the terms (apart from a list of stop words). Therefore, we propose the following strategy, when optimizing the model using Stochastic Gradient Descent (SGD). For the first 5 iterations, we omit the weighting function. After 5 iterations, we can evaluate for each term t how well the corresponding hyperplane is able to separate entities that have the term in their bag-of-words representations from the other ones. For the next 5 iterations, we set $\sigma(t)$ equal to the classification accuracy of this hyperplane. Finally, after each 5 iterations, we re-evaluate the performance of the hyperplanes and update the weighting function.

Finally, note that in the full model, the entity representations are regularized by the nuclear norms in J_{reg}. In variants of our model without this latter component, we will instead add L2 regularization.

3.3 Ordinal regression based embeddings

The model proposed in Section 3.2 is aimed at finding embeddings that can be used to verifying which entities have a given property. Many properties are a matter of degree, however, so it may often be more natural to model the extent to which each entity has a given property. In the following, we will use the PPMI measure to estimate how much each term is related to each entity ($e \in \mathcal{E}, t \in W$), defined as $PPMI(e,t) = \max(0, PMI(e,t))$ with

$$PMI(e,t) = \log \frac{n(e,t) \cdot (\sum_{e' \in \mathcal{E}} \sum_{t' \in W} n(e',t'))}{(\sum_{e' \in \mathcal{E}} n(e',t)) \cdot (\sum_{t' \in W} n(e,t'))}$$

where we write $n(e,t)$ for the number of times word t occurs in W_e, i.e. the bag-of-words representation of entity e. While there are other ways in which we can quantify how strongly a given term is related to an entity, PPMI is by far the most popular choice in the context of word embedding [41]. Note that, similar as in Section 3.2, we are assuming that the bag-of-words representations of the entities reflect the properties they have. Here we are additionally assuming that the more a property applies to an entity, the more it will be mentioned in textual descriptions of that entity. While this may seem like a strong assumption, we can indeed expect that adjectives such as 'tall' will mostly be used in the context of the tallest mountains, thus allowing us to distinguish exceptional elements (w.r.t. tallness). In addition, there will be words such as 'snow', 'top, or 'cloud' which may appear more proportionally, and could thus allow us to differentiate between other mountains.

For each term t, we consider a partition $\mathcal{E}_0^t, ..., \mathcal{E}_{m_t}^t$ of $\mathcal{E}$, such that (i) for every $e \in \mathcal{E}_0^t$ it holds that $PPMI(e,t) = 0$, and (ii) for every $e \in \mathcal{E}_i^t$ and $f \in \mathcal{E}_j^t$ with $i < j$, it holds that $PPMI(e,t) < PPMI(f,t)$. In practice, we may either choose the number of partition classes equal to the number of different PPMI-values (i.e. such that all elements of $\mathcal{E}_i^t$ have the same PPMI value), or we may consider a fixed number of bins. Given the partition $\mathcal{E}_0^t, ..., \mathcal{E}_{m_t}^t$, we can again use a squared hinge loss to require that there should be a sequence of parallel hyperplanes in the vector space that separate each partition class $\mathcal{E}_i^t$ from the previous one $\mathcal{E}_{i-1}^t$:

$$J_{text}^{reg} = \sum_{t \in W} \sum_{i=1}^{m_t} \frac{\sigma(t)}{|\mathcal{E}_t^{i-1}| + |\mathcal{E}_t^i|} \Big(\sum_{e \in \mathcal{E}_t^{i-1}} \max\left(0, 1 - (K(p_e, \tilde{w}_t) + b_t^i)\right)^2 + \sum_{e \in \mathcal{E}_t^i} \max\left(0, 1 + (K(p_e, \tilde{w}_t) + b_t^i)\right)^2 \Big) + \lambda \|\tilde{w}_t\|^2$$

This encoding is similar in spirit to the fixed margin variant for ranking with large-margin constraints proposed in [39], but with the crucial difference that we are again learning entity embeddings and hyperplanes at the same time, rather than finding hyperplanes for a given entity embedding. Another possibility, also suggested in [39], would be to minimize the sum of the margins separating the different partition classes for the same term. However, note that formulations such as the one proposed in [22], which are quadratic in the number of instances, would not be suitable here. When learning the space, following [9], we additionally impose the requirement that $b_t^1 < ... < b_t^{m_t}$ for every term t, i.e. that the parallel hyperplanes appear in the right order.

As the total number of entities in $\mathcal{E}_0^t$ would make the optimization problem too computationally demanding, we again only consider a sample of such negative examples. As before, the weighting function is chosen dynamically, with no weighting function considered for the first 5 iterations of the SGD algorithm. Instead of accuracy, we now choose $\sigma(t)$ as the Spearman ρ correlation between the ranking predicted by the parallel hyperplanes and the ranking induced by the partition $\mathcal{E}_0^t, ..., \mathcal{E}_{m_t}^t$.

3.4 Comparison with GloVe and EECS

In this paper, we keep the general formulation of the EECS model, shown in (1), but we change J_{text} by either J_{text}^{class} or J_{text}^{reg}. We also change J_{glove} in an analogous way. For example, if we replace J_{text} by J_{text}^{reg} with a quadratic kernel, we also model word-word co-occurrences using the ordinal regression model with a quadratic kernel (simply replacing in the formulation for J_{text}^{reg} the role of entity-word co-occurrences by word-word co-occurrences).

Similar to our max-margin based models, the GloVe based model in J_{text} learns a vector $\tilde{w}_t$ for every term. However, in general, this vector cannot directly be used to rank entities according to how closely they are related to term t, because of the inclusion of the bias term b_i. However, if we fix $b_i = \log(\sum_j y_{ji})$ it is easy to see that we can equivalently express J_{text} as follows:

$$J'_{text} = \sum_{e_i \in \mathcal{E}} \sum_{t_j \in W_{e_i}} f(y_{ji})(p_{e_i} \cdot \tilde{w}_j + b'_j - PMI(e_i,t))^2$$

This latter model is similar in spirit to our ordinal regression model with a linear kernel, in the case where each partition class $\mathcal{E}_i^t$ only contains elements with the same PPMI value. However, there are still two key differences: (i) entities e_i for which $t \notin W_{e_i}$ are ignored completely and (ii) the distances between the parallel hyperplanes are determined by the PMI values. The first difference seems to be problematic for the GloVe based model. Indeed, for rare (but informative) words in particular, it seems useful to model whether or not that word appears at least once in the bag-of-words representation of an entity. Regarding the second difference, it is unclear why PMI would be a good basis for determining the distances between the parallel hyperplanes. In the next section, we will experimentally compare what these differences mean in practice.

An advantage of our model, compared to EECS, is that evidence from bag-of-words representations can be treated in the same way as evidence coming from other sources. For example, suppose we know the height of a large number of mountains (e.g. from DBpedia), then we can include that evidence in the formulation J^{reg}_{text}, simply by considering a partition based on these numerical height values.

4 EVALUATION

4.1 Methodology

4.1.1 Data acquisition. Our model relies on bag-of-words representations obtained from Wikipedia. Following [23], we use WikiData as our source for structured information, including semantic types. In addition to making the comparison with [23] more straightforward, WikiData has the advantage of having a clean semantic type hierarchy, which is important given the central role that semantic types play in the model.

In our experiments, we have used the Wikipedia dump[2] from September 20, 2016. This dump contains about 5.2 million documents after removing the stub pages, disambiguation pages, etc. We pre-processed the Wikipedia dump by removing all HTML/XML tags, punctuations and non-ASCII characters, and we lower-cased all tokens. Sentence segmentation was done using the Apache sentence detector tool[3]. To generate the bag-of-words representation of a given entity, we have considered all the words appearing in the Wikipedia article about that entity, as well as the context words of mentions of that entity in other Wikipedia articles (choosing a window size of 10, but not crossing sentence boundaries). We obtain mentions of the entity from pages that contain a hyperlink to the Wikipedia article of that entity, considering the hyperlink itself as a mention, as well as all subsequent mentions of the name of that entity on the same page.

For WikiData, we used the JSON dump[4] from September 26, 2016. The WikiData dump was first indexed with MongoDB and relevant fields, such as entity IDs, aliases, Wikipedia title, relation information, and the type hierarchy information were then extracted from this database. For example, in order to extract which entity belongs to which type, we use the "instance-of" field. In order to determine the type hierarchy, we have used the "subclass-of" field. We will provide scripts to generate the full dataset online. In total, our dump of WikiData contains about 9 million entities. However, for many of these entities only very limited information is available, and for such entities, it is clearly not possible to learn reliable embeddings. Therefore, for our entity embedding, we only consider entities that are mentioned in at least 10 Wikipedia articles, leading to a collection of around 1.3 million entities.

4.1.2 Baseline Models. We have compared our models with several baselines and state-of-the-art models. In all cases, existing implementations were used. The most direct comparison is with the EECS model[5], which shares some components with our model and uses the same input. We have also compared with the pTransE model [49], which uses a combination of structured information and textual descriptions as well. We have experimented with different variants for the textual descriptions (e.g. only using the Wikipedia article itself, as proposed by the authors), but obtained the best results when using the same bag-of-words representations as for our model and the EECS model. This has the advantage that the results for our model, EECS and pTransE can be directly compared. Recall that pTransE essentially combines the Skip-gram model with a variant of TransE.

In addition, we also considered a number of models that only use the textual descriptions of the entities, including parametric[6] (Latent Dirichlet Allocation (LDA)) and non-parametric[7] (Hierarchical Dirichlet Processes (HDP)) latent topic models, the SVD implementation of Matlab, GloVe, Skip-gram, and CBOW. Conversely, we also considered a number of knowledge graph embedding models[8] (TransE, TransH, TransR, CTransR, RESCAL) which only use the structured information. We do not expect these models to be competitive (as they only have access to some of the information that other models have access to) but include them to evaluate what performance we can expect from textual information alone and from structured (WikiData) information alone.

4.1.3 Methodology. For the evaluation, we have used five-fold tuning, where the training set contained 60% of the data, and the testing set and tuning sets contained 20% and the data each. The tuning set was used for setting all parameters of the different models, such as the vector size, which we varied between 100 and 300. All results are averaged across five folds. The number of iterations for SGD was chosen as 20 for all models.

4.2 Results

4.2.1 WikiData experiments. In a first experiment, we evaluated the entity embeddings using two tasks that were proposed in the EECS paper [23]. The first task, called induction, consists of completing a list of entities that all share some (unknown) property. Specifically, when learning the entity embeddings some relations from WikiData were held out. Each problem instance consists of a list of entities that are related to some given entity (e.g. a list of movies that were all directed by James Cameron), according to this held-out part of WikiData. The aim is to find other entities that also have this property (e.g. other films directed by James Cameron). This task is treated as a ranking task. In particular, we rank all entities based on their distance in the space to the centroid of the given

[2] https://dumps.wikimedia.org/enwiki/20160920/
[3] https://opennlp.apache.org/documentation/manual/opennlp.html
[4] https://dumps.wikimedia.org/wikidatawiki/
[5] https://github.com/bashthebuilder/ECAI-2016
[6] http://www.cs.princeton.edu/ blei/lda-c/
[7] https://github.com/blei-lab/hdp
[8] https://github.com/thunlp/KB2E

Table 1: Evaluation of the entity embeddings on benchmark tasks derived from WikiData.

	Induction			Ranking
	MAP	P@5	MRR	ρ
TransE	0.158	0.202	0.456	0.178
TransH	0.158	0.211	0.419	0.166
TransR	0.159	0.302	0.489	0.176
CTransR	0.165	0.325	0.503	0.192
RESCAL	0.201	0.219	0.305	0.101
SVD	0.155	0.231	0.332	0.183
LDA	0.111	0.108	0.201	0.155
HDP	0.148	0.109	0.199	0.183
Skip-gram	0.172	0.355	0.502	0.154
CBOW	0.180	0.346	0.487	0.149
GloVe	0.221	0.569	0.823	0.259
pTransE	0.226	0.499	0.762	0.214
EECS	0.228	0.602	0.881	0.312
MEmbER(class,lin)	0.226	0.605	0.889	0.314
MEmbER(class,quad)	0.230	0.611	0.889	0.311
MEmbER(reg,lin)	**0.238**	**0.621**	0.895	0.319
MEmbER(reg,quad)	0.233	**0.621**	**0.899**	**0.341**

Table 2: Evaluation of the entity embeddings on word embedding benchmarks.

	Analogy	Outlier detection	
	Acc	Acc	OPP
TransE	42.7	61.6	89.8
TransH	41.8	65.9	90.1
TransR	42.6	65.4	90.0
CTransR	41.4	66.1	91.3
RESCAL	48.9	59.3	86.3
SVD	60.1	52.3	86.9
LDA	54.9	48.1	81.9
HDP	57.1	52.5	83.1
Skip-gram	69.8	69.7	93.2
CBOW	74.3	72.2	94.2
GloVe	78.8	67.8	92.9
pTransE	65.6	72.6	94.3
EECS	79.9	73.8	95.0
MEmbER(class,lin)	79.8	73.9	95.0
MEmbER(class,quad)	79.8	78.9	95.0
MEmbER(reg,lin)	80.2	79.0	95.0
MEmbER(reg,quad)	**80.9**	**79.1**	**95.1**

examples. The second task consists in completing a given ranking of entities. In each problem instance, we are given a ranking of some entities, according to some numerical attribute in WikiData (which was not used for learning the space), e.g. a list of mountains, ranked according to their height. The task is to rank other entities of the same semantic type, relative to the given ranking (e.g. to predict where other mountains belong in the ranking). To find this ranking, for a given vector space representation, we use RankSVM. For more details about this task, we refer to [23].

The results are summarized in Table 1. We can observe that the regression versions of our models consistently outperform EECS, as well as all the other baselines. In fact, even the MEmbER(class,lin) model, which simply finds a separating hyperplane for each term, performs comparably to EECS. This is surprising, given that all information about term frequency is ignored in this model, apart from whether a term occurs or not. In most cases, the quadratic kernel versions of our model perform slightly better than the linear kernel versions.

The pTransE model performs comparably to EECS and to the classification variants of our model for the induction task but is not competitive on the ranking task. As explained in Section 3.4, like our model, the GloVe based models could be interpreted as finding a space in which context words define rankings of entities. It seems plausible that this feature is crucial for solving ranking tasks. In the regression variants of our model, we make this ranking view more explicit, which could explain the gains we obtain over EECS.

Apart from GloVe, to some extent, the other baselines are not competitive. However, it is interesting to see that a method such as CTransR, which only uses the structured information can outperform baselines such as SVD, LDA, HDP, and can perform comparatively to Skip-Gram.

4.2.2 Word embedding benchmarks. While there are no standard benchmark datasets for evaluating entity embeddings, there are a number of datasets for evaluating word embeddings that can be reused, because all the words in these datasets are also Wikipedia entities. This is the case for the semantic Google Word analogy datasets[9] and for an outlier detection dataset[10]. The aim of the word analogy task is to complete analogical pairs of the form (*paris*, *france*) : (*london*, ?). The aim of the outlier detection task is to find words that do not belong in a given list. We refer to [8] for details about this latter task, including a motivation for the two evaluation metrics (which are both to be maximized).

The results of these two experiments are shown in Table 2. We again find that the regression variants of our models outperform EECS, as well as the other baselines and that the classification variants perform comparably to EECS. For the outlier detection task, the increase in accuracy is particularly noticeable. This suggests that our models are indeed better equipped to identify entities that have some given property (or in this case, have some property in common), which was an explicit design goal for our model. For the outlier detection task, it is interesting to see that CTransR can compete with the word embedding models, given that this task has been adapted from a word embedding evaluation task.

4.2.3 Web table completion. An important motivation for our models was the assumption that they would be better suited for ranking tasks. While we already evaluated a ranking task in Table 1, that task was limited in scope in the sense that it only considered rankings that were induced from numerical attributes. To further analyze the performance of our models for ranking tasks, we consider the task of predicting in which order entities appear in a given

[9] http://nlp.stanford.edu/projects/glove/
[10] http://lcl.uniroma1.it/outlier-detection/

Table 3: Evaluation of the entity embeddings for web table completion.

	$5 < n \leq 10$	$10 < n \leq 50$	$n > 50$
	Acc	ρ	ρ
TransE	0.624	0.301	0.313
TransH	0.615	0.303	0.233
TransR	0.634	0.305	0.342
CTransR	0.639	0.299	0.414
RESCAL	0.606	0.287	0.234
SVD	0.612	0.201	0.188
LDA	0.610	0.212	0.122
HDP	0.600	0.193	0.156
Skip-gram	0.650	0.676	0.460
CBOW	0.661	0.632	0.432
GloVe	0.667	0.663	0.459
pTransE	0.641	0.603	0.423
EECS	0.671	0.688	0.479
MEmbER(class,lin)	0.671	0.676	0.459
MEmbER(class,quad)	0.670	0.665	0.444
MEmbER(reg,lin)	0.691	0.699	0.499
MEmbER(reg,quad)	**0.693**	**0.702**	**0.501**

table. Specifically, we used the webtable dataset[11] from [25], which is a large collection of HTML tables extracted from web pages. We have considered only the relational tables, of which there are 90 million instances in the original dataset. The relational tables are particularly interesting for us because often each row of these tables corresponds to an entity. To find relevant tables, we selected tables in which the first column contained an increasing sequence of numbers, starting with 1, and in which the second column contained names of entities that could be mapped to Wikipedia entities. When mapping table entries to Wikipedia entities, we use the semantic type information from WikiData to resolve ambiguities, i.e if a given table entry could correspond to more than one entity, we assign it to the entity whose semantic type best matches that of the other table entries.

The assumption is that tables which satisfy the above conditions will usually mention the entities in some relevant order. We did not consider tables with 5 rows or less (after removing rows whose entity in the second column could not be mapped to our Wikipedia entities). For tables with 6 to 10 rows, we randomly remove two rows and consider the task of predicting in which order these rows appear in the table (using only the names of the entities in the second column of these rows). The evaluation metric, in this case, is accuracy. For tables with more than 10 rows, we remove a third of the rows and try to rank the corresponding entities. In this case, the evaluation metric is the Spearman ρ coefficient. We show results separately for tables of up to 50 rows and for tables with more than 50 rows. In total, we used 200,000 tables with 6 to 10 rows, 100,000 tables with 11 to 50 rows, and 25,000 tables with more than 50 rows. The resulting benchmark set will be made available online. To solve the task, we again apply RankSVM to the different vector space representations.

[11] http://webdatacommons.org/webtables/

The results are shown in Table 3. The regression variants of our model again consistently outperform EECS and the other baselines. In this task, the word embedding models (Skip-gram, CBOW and GloVe) perform surprisingly well, even outperforming pTransE, which uses structured information in addition to bag-of-words representations. It is also noticeable that the increase in the performance of EECS and our model is quite large for this task, relative to the differences between the other models (e.g. between GloVe and EECS). This lends further support to the view that our model is particularly well equipped to deal with ranking problems.

An interesting feature of our model is that we can use the vectors $\tilde{w}_t$ to find terms that describe the criteria underlying a given ranking. In particular, we can find terms t that describe these criteria, by comparing the vector learned by the RankSVM model with the vectors $\tilde{w}_t$ that were learned as part of the max-margin models, in terms of cosine similarity. For example, the webtables dataset contains a table about baseball players, listing (1) ichiro-suzuki, (2) hideki-matsui, (3) yu-darvish, (4) charlie-manuel, (5) sadaharu-oh, (6) hideo-nomo, (7) kazuo-matsui, (8) kosuke-fukudome, (9) kei-igawa, (10) kenji-johjima, (11) kenshin-kawakami, (12) alex-cabrera. Using our model, we identified the term "slugger" for this ranking, which is baseball parlance for "person who strikes hard".

4.2.4 Entity retrieval. In the last experiment, we used the data from the INEX entity retrieval tracks[12], where the aim was to rank entities given a text query and (in some variants) a few example entities that satisfy the query. The document collection considered in the INEX track was a 2009 Wikipedia dump, which is known as the INEX 2009 Wikipedia collection[13]. The topics (i.e. queries) and relevance assessments are also available online[14].

To use our entity embeddings for this task, we first mapped the document IDs in the relevance assessment files to those of our 2016 Wikipedia dump. Where possible, this mapping was based on the title fields of the Wikipedia articles. In the case of a mismatch, we used the aliases data to improve the accuracy of the matching algorithm. For the entities listed in the topics file of the INEX dataset (i.e. the example answers provided for each query), we manually verified the mapping. The reported numerical results have been obtained using the standard INEX Perl evaluation tool provided with the dataset. Our main retrieval engine is Terrier[15] with the Divergence from Randomness model as our baseline retrieval method with the default retrieval parameters implemented in Terrier.

We have conducted three different experiments using the INEX data. In the first experiment (referred to as *list*), we try to retrieve relevant entities by only using the example entities that were provided. This setting is similar to the induction experiment in Section 4.2.1, but with the crucial difference that here only 2 or 3 example entities are provided. For this experiment, we again rank entities based on their distance to the centroid of the given examples.

For the second experiment (referred to as *query*), we only used the text query. Following [33], we use a learning-to-rank approach to combine information from the vector space with the baseline retrieval model. Specifically, we use the title field of the INEX

[12] http://www.inex.otago.ac.nz/tracks/entity-ranking/guidelines.asp
[13] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/software/inex/
[14] http://inex.mmci.uni-saarland.de/data/documentcollection.html
[15] http://terrier.org/

topic and retrieve the top 10 entities using the baseline model (i.e. Divergence from Randomness). To obtain the final results, we then train a RankSVM model[16], which uses three features to re-rank the top 10 entities retrieved using the baseline:

- The ranking score from the Divergence from Randomness baseline.
- The distance of each entity to the centroid of the top-10 entities.
- The following score, which directly interprets the query in the vector space:

$$score(e) = \sum_{t \in Q} \sigma(t) \cdot pos(e) \quad (3)$$

where $\sigma(t)$ is the scoring function from the max-margin model (i.e. the accuracy or Spearman ρ coefficient associated with the vector $\tilde{w}_t$) and $pos(e)$ is the relative position of entity e in a ranking of all entities f according to $\tilde{w}_t$ (i.e. considering for each entity f the value $f \cdot \tilde{w}_t$ to determine the position of f in the ranking).

For the comparative models, the last component is omitted. For the third experiment, both the text query and the list of example entities is used. The set-up in this case is the same as in the second experiment, but we add the example entities to the top-10 entities obtained from the baseline.

The results are summarized in Table 4. For reference, using the Divergence from Randomness model alone leads to a MAP score of 0.0501 and NDCG score of 0.291. Our models here substantially outperform EECS, which outperforms the other models. In fact, most of these other baselines do not even succeed in improving the Divergence from Randomness model. Note that the overall MAP scores are very low, which is related to the fact that we did not use any category information. In particular, the INEX topics provide a Wikipedia category that loosely corresponds to the set of entities that should be returned. Obviously, this category information (together with a model that takes advantage of semantic relations between Wikipedia categories) is key for obtaining good results. Since the problem of effectively exploiting Wikipedia categories is orthogonal to our aims in this paper and has been extensively studied in previous work [2, 14, 24], we wanted to focus on how well the entity embeddings are able to interpret text queries and to generalize from a few examples.

The impact of the scoring function (3) was minimal. We believe that this is largely because the INEX queries focus on rather specific properties, whereas this scoring function would be most effective for salient properties of entities. For an example, if we consider the query "country population", using only the entity embedding, we obtain the following top-ranked entities: "china", "india", "america", "indonesia", "iran", "brazil". For "movies dinosaur" we obtain "dinosaur", "jurassicpark" as the top ranked entities, while for "france capital" we obtain "france", "capital", "paris".

5 CONCLUSIONS

Despite the popularity of word embedding and KG embedding methods, there is little existing work on the use of vector space embeddings for entity retrieval tasks. Moreover, while existing embedding methods mostly focus on modelling similarity, within the context of entity retrieval the most important tasks are to identify entities that have a given property, or to rank them according to how much they have that property. We have proposed a model based on max-margin constraints that directly encodes this requirement. It has the advantage that the resulting entity embedding is interpretable, in the sense that we can use the model to describe what features a given entity has, or conversely to retrieve the entities that are most strongly related to a given set of query terms.

As our experimental results have shown, despite its conceptual simplicity, our model consistently outperforms all of the state-of-the-art models, even on tasks that do not rely on having an interpretable embedding. This is remarkable, as interpretability usually comes at the price of lower performance on other metrics. For the considered evaluation tasks, we noted that combining structured information with textual descriptions is paramount to achieve good performance. Our model can combine these types of information in a natural way. Moreover, we can easily integrate other types of evidence, such as numerical attributes (which would simply give rise to another term in the model), or even ordinal inputs (e.g. using the order in which entities appear in web tables).

6 ACKNOWLEDGMENTS

This work was supported by ERC Starting Grant 637277. This work was performed using the computational facilities of the Advanced Research Computing@Cardiff (ARCCA) Division, Cardiff University. The authors would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. 2006. Formal models for expert finding in enterprise corpora. In *SIGIR*. 43–50.
[2] Krisztian Balog, Marc Bron, and Maarten De Rijke. 2010. Category-based query modeling for entity search. In *ECAI*. 319–331.
[3] Krisztian Balog, Arjen P. de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. 2009. Overview of the TREC 2009 Entity Track. In *TREC*.
[4] Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. 2010. Overview of the TREC 2010 Entity Track. In *TREC*.
[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR* 3 (2003), 1137–1155.
[6] Roi Blanco, Peter Mika, and Sebastiano Vigna. 2011. Effective and efficient entity search in RDF data. In *Proceedings of the International Semantic Web Conference*. 83–97.
[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 2787–2795.
[8] Jose Camacho-Collados and Roberto Navigli. 2016. Find the word that does not belong: A Framework for an Intrinsic Evaluation of Word Vector Representations. In *Proceedings of the ACL Workshop on Evaluating Vector Space Representations for NLP*.
[9] Wei Chu and S Sathiya Keerthi. 2005. New approaches to support vector ordinal regression. In *ICML*. 145–152.
[10] Marek Ciglan, Kjetil Nørvåg, and Ladislav Hluchỳ. 2012. The SemSets model for ad-hoc semantic list search. In *WWW*. 131–140.
[11] Arjen P De Vries, Anne-Marie Vercoustre, James A Thom, Nick Craswell, and Mounia Lalmas. 2007. Overview of the INEX 2007 entity ranking track. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*. 245–251.
[12] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.
[13] Gianluca Demartini, Arjen P de Vries, Tereza Iofciu, and Jianhan Zhu. 2008. Overview of the INEX 2008 entity ranking track. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*. 243–252.

[16] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Table 4: Evaluation of the entity embeddings on INEX entity retrieval tasks.

	List		Query		List + Query	
	MAP	NDCG	MAP	NDCG	MAP	NDCG
TransE	0.045	0.220	0.046	0.231	0.054	0.259
TransH	0.049	0.230	0.046	0.231	0.053	0.247
TransR	0.047	0.230	0.048	0.256	0.057	0.226
CTransR	0.044	0.234	0.046	0.256	0.057	0.226
RESCAL	0.031	0.211	0.033	0.218	0.022	0.229
SVD	0.041	0.221	0.045	0.236	0.046	0.237
LDA	0.039	0.213	0.040	0.223	0.033	0.257
HDP	0.039	0.214	0.041	0.226	0.045	0.212
Skip-gram	0.045	0.223	0.046	0.229	0.049	0.235
CBOW	0.043	0.213	0.049	0.215	0.049	0.219
GloVe	0.050	0.301	0.052	0.311	0.055	0.312
pTransE	0.040	0.253	0.049	0.301	0.059	0.301
EECS	0.058	0.338	0.059	0.339	0.059	0.339
MEmbER(class,lin)	0.064	0.348	0.065	0.349	0.065	0.349
MEmbER(class,quad)	0.064	0.343	0.065	0.343	0.065	0.349
MEmbER(reg,lin)	0.068	**0.352**	0.069	**0.353**	0.069	**0.359**
MEmbER(reg,quad)	**0.069**	**0.352**	**0.070**	0.352	**0.071**	**0.359**

[14] Gianluca Demartini, Claudiu S Firan, Tereza Iofciu, Ralf Krestel, and Wolfgang Nejdl. 2010. Why finding entities in Wikipedia is difficult, sometimes. *Information Retrieval* 13, 5 (2010), 534–567.

[15] Gianluca Demartini, Tereza Iofciu, and Arjen P De Vries. 2009. Overview of the INEX 2009 entity ranking track. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*. 254–264.

[16] J. Derrac and S. Schockaert. 2015. Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning. *Artificial Intelligence* (2015), 74–105.

[17] Jesse Dunietz and Daniel Gillick. 2014. A New Entity Salience Task with Millions of Training Examples. In *EACL*. 205–209.

[18] Maryam Fazel. 2002. *Matrix rank minimization with applications*. Ph.D. Dissertation. Stanford University.

[19] Besnik Fetahu, Ujwal Gadiraju, and Stefan Dietze. 2015. Improving entity retrieval on structured data. In *Proceedings of the International Semantic Web Conference*. 474–491.

[20] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word Embedding Based Generalized Language Model for Information Retrieval. In *SIGIR*. 795–798.

[21] P. Gärdenfors. 2000. *Conceptual Spaces: The Geometry of Thought*. MIT Press.

[22] R. Herbrich, T. Graepel, and K. Obermayer. 1999. Support vector learning for ordinal regression. In *ICANN*. 97–102.

[23] Shoaib Jameel and Steven Schockaert. 2016. Entity Embeddings with Conceptual Subspaces as a Basis for Plausible Reasoning. In *ECAI*. 1353–1361.

[24] Rianne Kaptein and Jaap Kamps. 2013. Exploiting the category structure of Wikipedia for entity ranking. *Artificial Intelligence* 194 (2013), 111–129.

[25] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *WWW*. 75–76.

[26] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *NIPS*. 2177–2185.

[27] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. 2181–2187.

[28] Xitong Liu, Fei Chen, Hui Fang, and Min Wang. 2014. Exploiting entity relationship for query expansion in enterprise search. *Information Retrieval* 17, 3 (2014), 265–294.

[29] Xitong Liu and Hui Fang. 2015. Latent entity space: a novel retrieval approach for entity-bearing queries. *Information Retrieval Journal* 18 (2015), 473–503.

[30] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proc. NAACL-HLT*. 746–751.

[31] David N Milne, Ian H Witten, and David M Nichols. 2007. A knowledge-based search engine powered by wikipedia. In *CIKM*. ACM, 445–454.

[32] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. 2012. On the Modeling of Entities for Ad-Hoc Entity Search in the Web of Data. In *ECIR*, Vol. 12. 133–145.

[33] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*. 809–816.

[34] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. 2016. Parameterized Fielded Term Dependence Models for Ad-hoc Entity Retrieval from Knowledge Graph. In *SIGIR*. 435–444.

[35] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.

[36] Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document retrieval using entity-based language models. In *SIGIR*. 65–74.

[37] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. 2004. A hybrid approach for searching in the semantic web. In *WWW*. 374–383.

[38] Michael Schuhmacher, Laura Dietz, and Simone Paolo Ponzetto. 2015. Ranking entities for web queries through text and knowledge. In *CIKM*. 1461–1470.

[39] Amnon Shashua and Anat Levin. 2002. Ranking with Large Margin Principle: Two Approaches. In *NIPS*. 937–944.

[40] Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux. 2012. Combining inverted indices and structured search for ad-hoc object retrieval. In *SIGIR*. 125–134.

[41] P. D. Turney and P. Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* 37 (2010), 141–188.

[42] Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*. 1591–1601.

[43] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. 1112–1119.

[44] Yang Xu, Gareth JF Jones, and Bin Wang. 2009. Query dependent pseudo-relevance feedback based on wikipedia. In *SIGIR*. 59–66.

[45] Hamed Zamani and W. Bruce Croft. 2016. Embedding-based Query Language Models. In *ICTIR*. 147–156.

[46] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *SIGIR*. 575–584.

[47] Nikita Zhiltsov and Eugene Agichtein. 2013. Improving entity search over linked data by modeling latent semantics. In *CIKM*. 1253–1256.

[48] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *SIGIR*. 253–262.

[49] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning Knowledge and Text Embeddings by Entity Descriptions. In *EMNLP*. 267–272.