Mapping Text to Knowledge Graph Entities
using Multi-Sense LSTMs

Dimitri Kartsaklis[*]   **Mohammad Taher Pilehvar**   **Nigel Collier**

Department of Theoretical and Applied Linguistics
University of Cambridge
Cambridge, UK
{dk426;mp792;nhc30}@cam.ac.uk

Abstract

This paper addresses the problem of mapping natural language text to knowledge base entities. The mapping process is approached as a composition of a phrase or a sentence into a point in a multi-dimensional entity space obtained from a knowledge graph. The compositional model is an LSTM equipped with a dynamic disambiguation mechanism on the input word embeddings (a *Multi-Sense LSTM*), addressing polysemy issues. Further, the knowledge base space is prepared by collecting random walks from a graph enhanced with textual features, which act as a set of semantic bridges between text and knowledge base entities. The ideas of this work are demonstrated on large-scale text-to-entity mapping and entity classification tasks, with state of the art results.

1   Introduction

The task of associating a well-defined action, concept or piece of knowledge to a natural language utterance or text is a common problem in natural language processing and generic artificial intelligence (Tellex et al., 2011), and can emerge in many different forms. In NLP, the ability to code text into an entity of a knowledge graph finds applications in tasks such as question answering and information retrieval, or any task that involves some form of mapping a definition to a term (Hill et al., 2016; Rimell et al., 2016). Further, it can be invaluable in providing solutions to domain-specific challenges, for example medical concept normalisation (Limsopatham and Collier, 2016) and identification of adverse drug reactions (O'Connor et al., 2014).

This paper details a model for efficiently mapping unrestricted text at the level of phrases and sentences to the entities of a knowledge base (KB)—a task also referred to as text *grounding* or *normalisation*. The model aims at characterising short focused texts, such as definitions or tweets. Given a medical KB, for example, a tweet of the form "Can't sleep, too tired to think straight" would be mapped to the entity *Insomnia*, while in the context of a lexical ontology the definition "Device that detects planets" would be associated to the entity *Telescope*.

Note that such a task cannot be approached as standard classification, since the "classes" (entities) are usually in one-to-one correspondence with the available inputs. To address this we propose the use of a continuous vector space for embedding the entities of the KB graph, where text is projected by a neural network. We rely on the notion of distributional semantics, where a word is represented as a multi-dimensional vector obtained either by collecting co-occurrence statistics with a selected set of contexts or by directly optimising an objective function in a neural network-based architecture (Collobert and Weston, 2008). Interestingly, similar techniques can be used for the multi-dimensional representation of nodes in a KB graph; for example, by collecting random walks following the edges of a graph it is possible for one to construct an artificial "corpus", to which a distributional model applies in the usual way (Perozzi et al., 2014).

By exploiting this representational compatibility, we treat the process of text-to-entity mapping as a transformation from a textual vector space where words live, to a KB vector space created from a graph and populated by vectors representing entities. A sentence is coded as a sequence of word vectors, composed by a modified Long Short-Term Memory network (LSTM—Hochreiter and Schmidhuber, 1997) into a multi-dimensional point in the entity space. One of our aims is to specifically deal with lexical ambiguity and polysemy which can be an important factor

---

[*] This paper is dedicated to the memory of Euripides Kartsaklis, a man who loved technology.

for the task at hand. To this end, each word is associated with a number of sense embeddings, and the LSTM is extended with an attentional disambiguation mechanism that dynamically selects and updates the right sense vector for each word given its context during training. We dub this formulation *Multi-Sense LSTM* (MS-LSTM).

An important issue is the provision of a set of reliable anchors; that is, points in one-to-one correspondence between the two representations that would enforce some degree of structural similarity between pieces of text and KB entities and thus make the mapping more efficient. We deal with this problem by extending the original KB graph with nodes corresponding to *textual features*, i.e. to words strongly associated to a specific entity and collected from various resources. A novel sampling strategy is detailed for incorporating these nodes to random walks, which are then fed to the skipgram model for producing an entity space. The results indicate that the textual nodes, being words *and* KB entities at the same time, do an extremely effective job in transforming the geometry of the entity space to the benefit of mapping the textual modality.

The proposed model is evaluated in three tasks: text-to-entity mapping on a dataset extracted from SNOMED CT[1], a medical knowledge base of 327,000 concepts; a reverse dictionary task based on WordNet (Miller, 1998), where the goal is to associate a multi-word definition to the correct lemma (Hill et al., 2016); and document classification on the Cora dataset (McCallum et al., 2000). The results demonstrate the effectiveness of our methods by improving the current state of the art.

2 Background

Aligning meaning between text and entities in a knowledge graph is a task traditionally based on heuristic methods exploiting text features such as string matching, word weighting, syntactic relations, or dictionary lookups (McCallum et al., 2005; Lu et al., 2011; O'Connor et al., 2014). Machine learning techniques have been also exploited in various forms, for example Leaman et al. (2013) use a pairwise learning-to-rank technique to learn the similarity between different terms, while Limsopatham and Collier (2015) apply statistical machine translation to "translate" social media text to domain-specific terminology. There is little work based on neural networks; the most relevant to us

is a study by Hill et al. (2016), who tested a number of compositional neural architectures trained to approximate word embeddings on a reverse dictionary task. Compared to their work, this paper proposes the use of a distinct target space for representing ontological knowledge, where every entity in the graph lives.

The goal of a graph embedding method is to embed components of a knowledge graph into a low-dimensional space. One research direction focuses on the relations, i.e. the edges of the graph (Bordes et al., 2013; Socher et al., 2013; Xiao et al., 2016) and aims at tasks such as link prediction and KB completion. Such work is outside the scope of the current paper, the subject of which is the efficient low-dimensional representation of *entities* (nodes). In this line of research, a prevalent method involves the collection of a set of random walks, starting from each node in the graph (Perozzi et al., 2014; Grover and Leskovec, 2016). There is a direct analogy between such a set of random walks and a text corpus: each node corresponds to a word and the sequence of nodes visited during a random walk is analogous to a sentence. Thus, any distributional model that takes as input this artificial "corpus" can generate multidimensional representations of the nodes in the graph. Random walks have also been used for KB inference (Lao et al., 2011) with success.

While random walk-based methods are not the only way to construct graph spaces—alternatives include factorisation (Ahmed et al., 2013) and deep autoencoders (Wang et al., 2016)—they have been found very effective in capturing multiple aspects of the graph structure (Wang et al., 2017; Goyal and Ferrara, 2017). The current paper proposes a random walk generation strategy that improves and complements existing approaches.

The idea of using textual features to improve the entity vectors is not well explored, and most of the existing work focuses again on the representation of relations (Xie et al., 2016; Wang et al., 2014; Wang and Li, 2016) as opposed to entities. Closer to us is the work of Yamada et al. (2017) and Yang et al. (2015), with the latter to incorporate text features in the concept embeddings by exploiting matrix factorisation properties.

Representing the meaning of words using a number of sense vectors is an old and well-established idea in NLP—see for example (Schütze, 1998; Reisinger and Mooney, 2010; Neelakantan et al., 2014). However, most of the relevant research is evaluated on intrinsic tasks
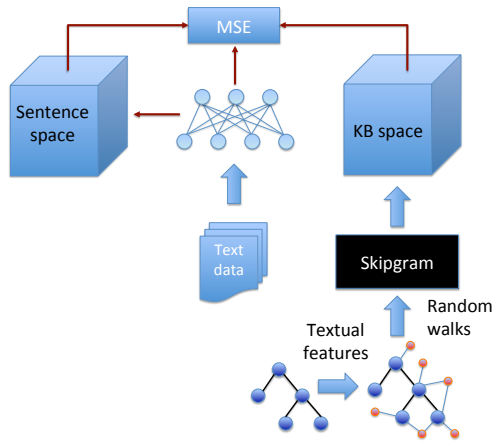

Figure 1: The text-to-entity mapping system in a nutshell. The red nodes indicate textual features, while "MSE" stands for mean squared error.

such as word similarity, while in the few works based in real end tasks, disambiguation is usually treated as a prior stand-alone step (Kartsaklis and Sadrzadeh, 2013; Li and Jurafsky, 2015; Pilehvar et al., 2017). The crucial difference of this work is that the ambiguity resolution mechanism is part of the compositional model itself, and the sense embeddings are trained simultaneously with the rest of the parameters. A close work is by Cheng and Kartsaklis (2015), who used a siamese network with an integrated disambiguation mechanism for paraphrase detection. For more information on multi-sense embeddings see (Camacho-Collados and Pilehvar, 2018).

3 Methodology

Fig. 1 provides a high-level illustration of our methodology, consisting of two stages: (1) the KB graph is extended with weighted textual features, and an artificial "corpus" of random walks is created and used as input to the skipgram model (Mikolov et al., 2013) for generating an enhanced KB space—this part is covered in §3.1; (2) the transformation from text to entities is performed by a supervised multi-sense compositional model, which generates a point in the KB space for every input text. This is achieved with an LSTM recurrent network, equipped with an attentional mechanism that provides a finer level of granularity to the different ways a word is used in the data—we detail this part in §3.2.

3.1 Textual features for entity vectors

For our KB space, we follow the generic recipe proposed by Perozzi et al. (2014) and we assemble

an artificial corpus of random walks from the KB graph, which is then used as input to the skipgram model (Mikolov et al., 2013). For a random walk of nodes $n_1 n_2 \ldots n_T$ and a context window size c, skipgram maximises the following quantity:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(n_{t+j}|n_t) \qquad (1)$$

i.e. for a target node n_t, the objective is to predict all other nodes in the same context. As a consequence, two vectors of the resulting space will be close if their corresponding nodes occur in topological proximity within the graph. However, while such a topology allows perhaps for meaningful comparisons between points in this space, it is not directly compatible with the task of mapping text to entities. The reason is that the communities formed in a KB graph (and thus the topology of the resulting vector space) mostly reflect domain-specific hierarchies and ontological relationships that are not necessarily evident by the textual representations referring to the entities. An important question therefore with regard to the proposed methodology is how to provide meaningful links between the two representations that would allow for the efficient translation of one form (text) to another (entities).

In this work, we address this problem by associating every node in the graph with a set of textual features, each one of which is weighted according to their importance with respect to the node. Our methodology is as follows: For each entity, we collect all available textual descriptions found in the knowledge base itself and the English portion of BabelNet (Navigli and Ponzetto, 2012), which is a very large dictionary integrating numerous resources, such as WordNet, Wikipedia, FrameNet and many others. The textual descriptions are treated as short documents, and each word in them is assigned a specific TF-IDF value, forming the set of textual features for the specific entity.

The KB graph is extended in the following way: Let T_c be the set of textual features for an entity c; then, for each t in T_c, we add an edge (c, t) with weight tf-idf$_c(t)$, where tf-idf$_c(t)$ is the TF-IDF value of t with respect to c. In contrast to Perozzi et al. (2014) who utilise a uniform node sampling strategy, we define the random walk generation process as follows: Given a randomly selected node n, let $C_n = \{c_1, c_2, \cdots c_N\}$ be the set of all entity nodes in its immediate vicinity, and $T_n = \{t_1, t_2, \cdots t_M\}$ the set of all textual features
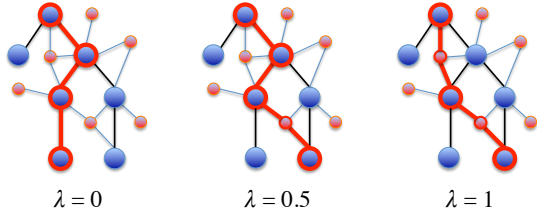
Figure 2: Effect of λ parameter. Blue nodes indicate entities, red nodes textual features, and red paths refer to random walks. As λ increases, the probability of "hops" between originally unlinked nodes increases accordingly.

of n; the next node x in the path is drawn from a categorical distribution defined as below:

$$
P_X(x) = \begin{cases} (1-\lambda)\dfrac{1}{N} & \text{if } x \in C_n \\[2ex] \lambda\dfrac{\text{tf-idf}_n(t_i)}{\sum_{j=1}^{M}\text{tf-idf}_n(t_j)} & \text{if } x \in T_n \end{cases} \quad (2)
$$

for X a discrete random variable with range $C_n \cup T_n$. In the above, λ defines the proportion of the probability mass allocated to textual features, when both C_n and T_n are non-empty; if one of the sets is empty, all of the probability mass is allocated to the other set, and λ becomes irrelevant. Further, in contrast to what is the case for the textual nodes, the probabilities of the entity nodes in Equation 2 are defined uniformly, since we lack any mechanism for fine-tuning them in a way that objectively reflects the importance of the nodes.

It is instructive to examine how the above sampling strategy works. As expected, setting $\lambda = 0$ will result in a sampling process that ignores the textual features and produces a path comprised solely of entity nodes; this is equivalent to the original model by Perozzi et al. (2014), known as DeepWalk. On the other hand, the effect of setting $\lambda = 1$ is less intuitive: Recall that, by construction, each textual node is connected only to entity nodes; that is, when the current node is textual, the next node will be always sampled from C_n. Therefore, setting $\lambda = 1$ creates paths following an alternating pattern, where each entity node is followed by a textual node, which in turn is followed by an entity node. Values of λ between 0 and 1 scale this behaviour accordingly (Fig. 2).

Advantages. The introduction of textual features in the graph achieves two goals. Firstly, the textual nodes serve as links between entities which, although perhaps related to each other in some way, lie in different parts of the KB graph (e.g. being parts of different hierarchies). As
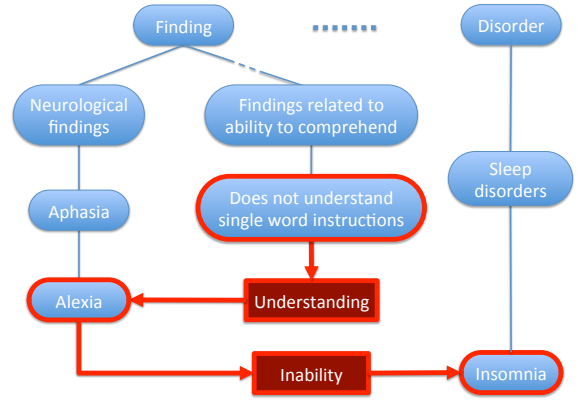

Figure 3: Linking of distant concepts with textual features (red boxes) for $\lambda = 1$. The textual feature *understanding* correctly links a related medical finding (lying on a different branch) to the condition known as *alexia*. Further, due to the presence of *inability* in their contexts, the vectors of *alexia* and *insomnia* (concepts originally quite far apart in the graph) will now have a common part reflecting that they are both conditions related to forms of incompetence.

a result, points that would normally be unjustifiably apart of each other in the vector space are now brought closer, providing additional coherence. This behaviour is controlled by the λ parameter, as Fig. 2 shows. Fig. 3 presents an illustrative example, taken from a real random walk on SNOMED CT.

The second advantage of introducing textual features in the graph is a consequence of the dual nature of these features in the context of learning: they essentially represent words, but since they are also nodes of the graph, they get vector representations exactly as every other normal entity in the knowledge base. The textual features, therefore, paired with their assigned vectors, form a set of *anchors* that links pieces of text with the KB space, and can be used to support the training process of the mapping system. In §4.1 we will see that this approach leads to substantial improvements in the accuracy of the model.

3.2 A multi-sense LSTM

We now proceed to present our neural architecture for text-to-entity mapping. The goal of the model is, given a certain piece of text, to produce a point in the KB space corresponding to an appropriate entity or concept. The model is trained on pairs of texts and entity vectors created from a graph extended with textual features, as discussed in §3.1.

Our architecture needs to explicitly take into account the fact that the task at hand is very sen-
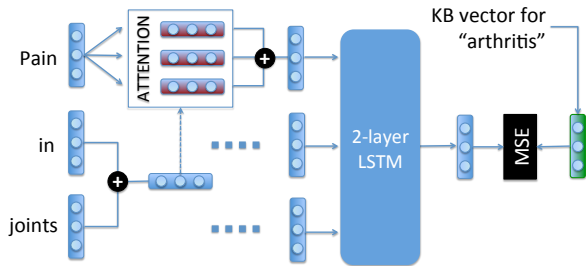
Figure 4: Detailed architecture of the Multi-Sense LSTM (MS-LSTM), shown here for the first word of a phrase. Red vectors refer to senses, while the green vector is a target vector.

sitive to lexical ambiguity. Specifically, while it is true that the level of homonymy (words having more than one disjoint meanings) is substantially decreased when we move from the generic domain to more specialised domains, on the other hand the increase in polysemy (words with many slightly different meanings) is exponential. As an example, while the lemma for the word "fever" in a dictionary usually contains two or three definitions, the term occurs in many dozens of different forms and contexts in SNOMED. Note that most of the different uses of the term correspond to distinct KB nodes, a fact that makes the job of a text-to-entity mapping system especially hard.[2] This motivates the employment of a dedicated mechanism that would handle the extra complexity imposed by the polysemous words.

The compositional setting of this paper, equipped with such a mechanism, is shown in Fig. 4. It consists of a generic word embedding layer, a word sense disambiguation layer, and two consecutive LSTM networks responsible for encoding the embeddings into a vector in the KB space. The objective is to minimise the mean squared error between the predicted vectors and the target vectors (prepared as in §3.1):

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \|y_i - f(x_i)\|^2 \qquad (3)$$

where N is the number of training examples, x the input text, y the target entity vector, and f the neural network.

To address the polysemy issues discussed above, every word is associated with a single generic embedding and k sense embeddings, where k is a fixed number. These sense embeddings can be seen as centroids of clusters denoting

---

[2]See also §4.4 for some concrete examples.

different uses of the word in the training set, and are dynamically updated during training. Specifically, for each word w_i in a training example, a context vector c_i is computed as the average of the generic vectors of all other words in the sentence. The probability of each sense vector s_{ij} given this context is then calculated via an attentional mechanism equipped with a softmax layer, as follows:

$$p(s_{ij}|c_i) = \frac{\exp\left(w_j'^{\mathsf{T}} s_{ij}'\right)}{\sum_{l=1}^{k} \exp\left(w_l'^{\mathsf{T}} s_{il}'\right)} \qquad (4)$$

where $s_{ij}' = \tanh(W s_{ij} + U c_i)$, and W, U and W' the parameters of the attentional network. Each sense vector is subsequently updated by addition of the context vector weighted by its similarity with the specific sense:

$$s_{ij}^{t+1} = s_{ij}^t + (s_{ij}^t{}^{\mathsf{T}} c_i^t)c_i^t \qquad (5)$$

The output of the attention is a weighted sum of the sense vectors given their probabilities (i.e. we apply *soft attention*), which is used as input to the compositional network—a 2-layer LSTM. The overall model is optimised on the MSE of the LSTM's output vector and the target entity vector. At inference time, a predicted vector $\hat{y}$ can be classified to the entity with the closest vectorial representation according to some metric.

4 Experiments

The ideas presented in the previous sections are evaluated on three tasks, two of which are related to text-to-entity mapping, and one to classification of KB entities. The purpose of the classification task (§4.3) is to provide a direct comparison of the textually enhanced vectors against vectors produced by the original graph, but *independently* of the compositional part. On the other hand, the text mapping experiments (§§4.1, 4.2) evaluate the overall architecture of Fig. 1 (including the compositional model and the dynamic disambiguation mechanism) on appropriate end tasks. Comparisons are provided with the most relevant previous work. Specifically, in all tasks, no inclusion of textual features corresponds to the standard DeepWalk model of Perozzi et al. (2014); in §4.2 our compositional architecture is compared to the work of Hill et al. (2016) in their reverse dictionary task; and §4.3 compares our method for textually enhancing the entity space with that of Yang et al. (2015), and other state-of-the-art deep models. The last subsection, §4.4, examines a few selected cases from a qualitative perspective.

4.1 Text-to-entity mapping

We begin with a large scale text-to-entity mapping experiment. We construct a dataset of 21,000 medical concepts extracted from SNOMED CT, each of which is associated with a multi-word textual description, taken from the knowledge base or BabelNet. The criterion for including a concept in the dataset was the availability of at least one textual description with 4 or more words. The objective of the task is to associate each one of these descriptions to the correct concept. Given a predicted vector $\hat{v}$, we assemble a list of all candidate concept vectors ranked by their cosine similarity with $\hat{v}$. We compute strict accuracy (based on how many times the vector of the correct concept is at the top of the list) and accuracy on the first 20 elements of the list. Further, we also present results based on the mean reciprocal rank (MRR).

In all experiments, we create KB vectors of 150 dimensions by applying the skipgram objective on a set of random walks of length 20, and with window size of 5. The graph is extended with 102,500 textual nodes weighted by their TF-IDF values with regard to the corresponding entities and selected as described in §3.1 (textual features that occur in the testing set are not taken into account). Each node in the graph serves as the starting point of 10 random walks. For the compositional model, we use embeddings of 150 dimensions, and 200-dimensional hidden states. The attentional mechanism is implemented as a 2-layer MLP, with 50 units allocated to the hidden layer for each sense. The overall model contains two dropout layers for regularisation purposes, and is optimised with Adam (Kingma and Ba, 2015) ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$).[3]

Following usual practice, we split our dataset in three parts: a training set (14,754 instances), a testing set (4,187 instances), and a development set (2,000 instances). We use the dev set to optimise the two main hyper-parameters of our model, namely the probability mass given to textual features (λ) and the number of senses for each word (k). The experiments on the dev set showed that increasing the probability mass for the inclusion of textual features in the random walks leads to consistently better performance for all tested models, so for the main experiment we set λ to its highest possible value, 1.00.[4] Further, a number of senses

Model	Target space	MRR	Acc	Acc-20
Baseline 1	W2V-GoogleNews	0.25	0.19	0.41
Baseline 2	W2V-PubMed	0.17	0.12	0.31
Least squares	DeepWalk	0.19	0.10	0.49
	TF vectors	**0.49**	**0.37**	**0.79**
CCA	DeepWalk	0.36	0.24	0.70
	TF vectors	**0.71**	**0.60**	**0.94**
Standard LSTM (150 dim.)	DeepWalk	0.30	0.20	0.58
	TF vectors	**0.82**	**0.73**	**0.97**
Standard LSTM ($k \times 150$ dim.)	DeepWalk	0.33	0.23	0.59
	TF vectors	**0.86**	**0.80**	**0.97**
MS-LSTM	DeepWalk	0.36	0.26	0.60
	TF vectors	**0.89**	**0.84**	**0.98**
MS-LSTM + anchors	TF vectors	<u>**0.94**</u>	<u>**0.90**</u>	<u>**1.00**</u>

Table 1: Results for the SNOMED dataset. For the MS-LSTM we set $k = 3$, while *TF vectors* refers to our textually enhanced vectors ($\lambda = 1$). The difference between MS-LSTM and LSTM is s.s. with $p < 0.01$ according to a two-tailed z-test.

equal to 3 achieved the highest performance.

We compare our MS-LSTM with a number of baselines: In *Baselines 1* and *2* a vector for each textual description is computed as the average of pre-computed word vectors, and compared to concept vectors prepared in a similar way, i.e. by averaging pre-computed vectors for all words in the qualified name of the entities. We used two different word spaces, a standard Word2Vec space created from Google News[5] and a custom Word2Vec model trained on a corpus of 4B tokens from medical articles indexed in PubMed[6]. In *Least squares* and *CCA*, an averaged vector for each textual description is again computed as before, and a linear mapping is learned between the textual space and the KB space, using least squares and canonical correlation analysis (Hardoon et al., 2004).

In *Standard LSTM*, we use a configuration similar to that of Fig. 4, but *without* the multi-sense aspect; here, the word embeddings are just parameters of the model randomly initialised before training. Further, we also test a standard LSTM where the length of the single embeddings is k times bigger (k is the number of senses in the MS-LSTM), so that the overall dimensionality of embeddings in LSTM and MS-LSTM is the same.

The results are presented in Table 1. Each model is tested against two target KB spaces, one consisting of simple DeepWalk vectors[7] and one of textually enhanced vectors (TF vectors, $\lambda = 1$) according to the procedure of §3.1. There are three observations: (1) Using the enhanced vectors as a

---

[3]Python code will be released at https://github.com/cambridgeltl/SIPHS.

[4]Recall that this means half of the nodes in a random walk will be textual (see Fig. 2).

[5]https://code.google.com/archive/p/word2vec

[6]http://www.ncbi.nlm.nih.gov/pubmed/

[7]In our setting, this is equivalent to having $\lambda = 0$.

target space improves the performance of *all* tested models by a large margin; (2) the MS-LSTM configuration of Fig. 4 achieves the highest overall performance, showing that explicitly handling polysemy during the composition is beneficial for the task at hand; and (3) despite the equal dimensionality between the two models, the standard LSTM with the long embeddings presents performance inferior to that of the MS-LSTM.

The last row of the table presents results after extending the training dataset with the textual anchors, that is, all the textual features paired with their learned KB vectors, as described in the *Advantages* section in §3.1. Specifically, recall that each textual feature (a word or a two-word compound), being also a node in the graph, is associated with a vector according to the process of §3.1. It is possible for one then to use these (*textual feature*, *vector*) pairs as additional examples during the training of the MS-LSTM. The last row of Table 1 shows the results after extending the training set with the 102,500 textual features. This setting achieves the highest performance, increasing further the strict accuracy by 6%, to 0.90.

4.2 Reverse dictionary

We proceed to the reverse dictionary task of Hill et al. (2016), the goal of which is to return a candidate term given a definition. Many forms of this task have been proposed in the past, see for example (Kartsaklis et al., 2012; Turney, 2014; Rimell et al., 2016). In (Hill et al., 2016), the authors test a number of supervised models under two evaluation modes: (1) "seen", in which the testing instances are also included in the training set; and (2) "unseen", where the evaluation is done on a held-out set. In both cases the datasets consisted of 500 term-definition pairs from WordNet.

We treat WordNet as a graph, the edges of which are defined by the various relationships between the synsets. This graph is further extended with 96,734 textual nodes extracted from the synset descriptions. We compute synset vectors of 150 dimensions, on random walks of length 20 and with window size of 5. For the seen evaluation, we train the compositional model on the totality of WordNet 3.0 synsets (117,659) and their descriptions. For the unseen evaluation, we remove from the graph any textual features occurring in the testing part, and create a new set of synset vectors; further, any testing instance is removed from the training set of the compositional model. The evaluation is done by comparing the

Model	Acc-10	Acc-100
Seen (500 WordNet definitions)		
OneLook (Hill et al., 2016)	0.89	0.91
RNN cosine (Hill et al., 2016)	0.48	0.73
Std LSTM (150 dim.) + TF vec.	0.86	0.96
Std LSTM ($k \times 150$ dim.) + TF vec.	0.93	0.98
MS-LSTM +TF vectors	**0.95**	**0.99**
MS-LSTM +TF vectors + anchors	**0.96**	**0.99**
Unseen (500 WordNet definitions)		
RNN w2v cosine (Hill et al., 2016)	0.44	0.69
BOW w2v cosine (Hill et al., 2016)	0.46	0.71
Std LSTM (150 dim.) + TF vec.	0.72	0.88
Std LSTM ($k \times 150$ dim.) + TF vec.	0.77	0.90
MS-LSTM + TF vectors	**0.79**	**0.90**
MS-LSTM + TF vectors + anchors	**0.80**	**0.91**

Table 2: Results for the reverse dictionary task, compared with the highest numbers reported by Hill et al. (2016). *TF vectors* refers to textually enhanced vectors with $\lambda = 1$. For the MS-LSTM, k is set to 3.

predicted vector with the vectors of all WordNet synsets (a search space of 117,659 points) and creating a ranked list as before, by cosine similarity. Following (Hill et al., 2016), we compute accuracy on top-10 and top-100. λ and k are tuned on a dev set of 2,000 synsets, showing a behaviour very similar to that of the SNOMED task.

Table 2 shows the results, based on a MS-LSTM setup similar to that of §4.1. Note that the MS-LSTM achieves 0.95-0.96 top-10 accuracy for the seen evaluation, significantly higher not only than the best model of Hill et al. (2016), but also higher than OneLook, a commercial system with access to more than 1000 dictionaries. It also presents considerably higher performance in the unseen evaluation. We are not aware of any other models with higher performance on the specific task.

4.3 Document classification

Our last experiment is a document classification task, performed on Cora (McCallum et al., 2000), a dataset containing 2708 machine learning papers linked by citation relationships into a graph. Each document is a short text extracted from the title or the abstract of the paper. The task is to predict the category of a document (a total of 7 classes), given its vector—so here we only evaluate the textually enhanced vectors as inputs to a classifier, independently of the compositional part.

In Table 3 we report results for two evaluation settings. In Evaluation 1, we provide a comparison with the method of Yang et al. (2015) who include textual features in graph embeddings based on matrix factorisation, and two topic models used as baselines in their paper. Using the same clas-

Model	Accuracy
Evaluation 1 (training ratio=0.50)	
PLSA (Hofmann, 1999)	0.68
NetPLSA (Mei et al., 2008)	0.85
TADW (Yang et al., 2015)	0.87
Linear SVM + DeepWalk vectors	0.85
Linear SVM + TF vectors	**0.88**
Evaluation 2 (training ratio=0.05)	
Planetoid (Yang et al., 2016)	0.76
GCN (Kipf and Welling, 2017)	0.81
GAT (Veličković et al., 2018)	**0.83**
Linear SVM + DeepWalk vectors	0.72
Linear SVM + TF vectors	**0.82**

Table 3: Results for the Cora dataset. *TF vectors* refers to textually enhanced KB vectors ($\lambda = 0.5$). Difference between our best models and GAT/GCN/TADW are *not* s.s.

sification algorithm (a linear SVM) and training ratio (0.50) with them, we present state-of-the-art results for vectors of 150 dimensions, prepared by a graph extended with 1422 textual features. We set $\lambda = 0.5$ by tuning on a dev set of 677 randomly selected entries from the training data.[8]

In Evaluation 2, using the same linear SVM classifier and λ as before, we reduce the training ratio to 0.05 in order to make our task comparable to the experiments reported by Veličković et al. (2018) for a number of deep learning models: specifically, the graph attention network (GAT) of Veličković et al. (2018), the graph convolutional network (GCN) of Kipf and Welling (2017), and the Planetoid model of Yang et al. (2016). Again, our simple setting presents results within the state of the art range, comparable to (or better than) those of much more sophisticated models that have been specifically designed for the task of node classification. We consider this as a strong indication for the effectiveness of the textually enhanced vectors as representations of KB entities.

Fig. 5 provides a visualisation of the Cora classes based on node vectors created with $\lambda = 0$ and $\lambda = 0.5$, correspondingly, demonstrating the impact of textual features in terms of cluster coherence and separation.

4.4 Qualitative evaluation

Table 4 compares the performance of the multi-sense approach with that of the single-sense model for a number of selected cases of text mapping. The predictions in the top part (for definitions

---

[8]We also attempted a second classification experiment on a dataset of 200k concepts extracted from SNOMED, observing a similar behaviour of λ (details are not reported due to space). This difference in the behaviour of λ between text-to-entity mapping and classification tasks is discussed in §5.
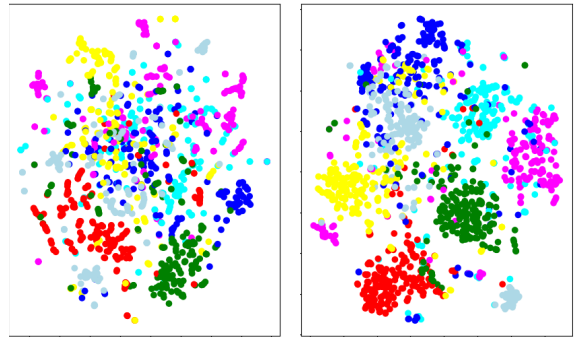

Figure 5: Visualisation of the Cora classes based on a 2D t-SNE projection of the node vectors before the inclusion of textual features (left) and after (right).

taken from the unseen evaluation of the reverse dictionary task) show that, in contrast to the single-sense model, the multi-sense approach was able to capture subtle variations of meaning between different synsets due to polysemy, as motivated in §3.2. The lower part of the table contains short phrases with ambiguous words, specifically selected to demonstrate the effect of the multi-sense approach. In all these cases, the multi-sense model was able to effectively disambiguate the ambiguous parts of the phrase by using the available context, and predict a very relevant synset; in contrast, the predictions of the single-sense model were based on choosing a wrong sense.

Finally, Table 5 presents the derived senses for word *table*, expressed as lists of nearest neighbouring words in the space. The model was able to effectively distinguish between a table as a kitchen furniture (sense 2), and a table as a structured way of presenting data (senses 1 and 3).

5 Discussion

The experimental work shows that using a graph embedding space as a target for mapping text to entities is an effective approach. This was mostly evident in the reverse dictionary task of §4.2, where the model was found to perform substantially better than previous approaches by Hill et al. (2016), who used a compositional architecture similar to ours but optimised on the *word* embeddings of the target terms. Note that this is suboptimal in the sense that, unless specific measures are taken, a word embedding reflects ambiguous meaning; therefore, trying to associate a definition like "keyboard musical instrument with pipes" to the vector for word "organ" introduces a certain amount of noise in the model, since the definition will be partly associated with features related to

Definition from the unseen dataset of the reverse dictionary task	$k = 3$ (correct pred.)	$k = 1$ (wrong pred.)
the branch of engineering that deals with things smaller than 100 nm	nanotechnology	microelectronics
floor consisting of open space at the top of a house just below roof	loft	balcony
a board game for two players; pieces move according to dice throws	backgammon	checkers
an address of a religious nature	sermon	rogation
Example short phrase with ambiguous words	$k = 3$ prediction	$k = 1$ prediction
a rechargeable cell	nickel-cadmium battery	karyolysis (*biological process*)
a state capital	Curitiba (*Brazilian state capital*)	assert (*verb*)
the lap of a person	upper side of thighs	lapper (*garment*)
a band named Queen	band leader	neckband (*garment*)

Table 4: Qualitative comparison of a few selected cases for multi- and single-sense LSTMs.

Sense 1. formulation, uncommonly, rauwolfia, cardiology, hypodermic, malleability, points, optic, dendrite, rubiaceae, nonparametric, meninges, deviation, anesthetics
Sense 2. tableware, meal, expectation, heartily, kitchen, hum, eating, forestay, suitors, croupier, companionship, restaurant, dishes, candles, cup, tea
Sense 3. reassigned, projective, ultracentrifuge, polemoniaceous, thyronine, assumptions, lymphocyte, atomic, difficulties, intracellular, virgil, elementary, cartesian

Table 5: Derived senses for word *table*, visualised as lists of nearest neighbouring words in the vector space.

the "body part" sense of the word. In our model, homonymy issues are resolved by design: each point in the target space corresponds to a well-defined unambiguous concept or synset. Further, the attentional mechanism of Fig. 4 handles subtle variations of each distinct sense due to polysemy.

The effectiveness of the textual feature mechanism was demonstrated in every task we attempted, but to different extents. As our tuning on the dev sets showed, for tasks closer to text-to-entity mapping (§§4.1-4.2) the more the textual features in the random walks, the better the results were. However, the best performance on the classification task came by λ values between 0.50 and 0.75, i.e. by walks visiting more entity nodes than textual nodes. The reason is that entity classification is a task very sensitive to the topology of the KB graph, since entities belonging to a specific class are very likely to be located at the same sub-hierarchy, hence in topological proximity. On the other hand, one of the motivations for introducing textual features was exactly to broaden the context of a node by connecting distant parts of the graph (see Figures 2-3). So, while small amounts of textual features can be still useful for classification purposes, excessive use introduces unwanted noise in the model.

The dynamic disambiguation mechanism integrated in the compositional architecture improved further the performance of the model. This finding is consistent with previous work on simpler tensor-based models, which showed that applying some

form of word sense disambiguation when composing word vectors can provide consistent improvements on end tasks such as sentence similarity and paraphrase detection (Kartsaklis and Sadrzadeh, 2013; Kartsaklis et al., 2013).

6 Conclusion and future work

We presented and evaluated a text-to-entity mapping system based on a continuous KB space enhanced with textual features and capable of handling polysemy. The reasonable next step will be to extend our methods for modelling the *relations* (edges) of a KB graph, which will allow applications in tasks such as link prediction and KB completion. Furthermore, having a mechanism that translates arbitrary text to points in a continuous space creates many opportunities for interesting research. For example, while the size of a knowledge base is finite, the space itself consists of infinite number of points, each one of which corresponds to a valid—yet not explicitly stated in the KB—entity of the same domain. The exciting question of how can we exploit this extra information—for instance in order to *enrich* the knowledge base with new data—constitutes one of our future directions.

References

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 37–48, New York, NY, USA. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Jose Camacho-Collados and Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. To appear in *Journal of Artificial Intelligence Research* (JAIR).

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Palash Goyal and Emilio Ferrara. 2017. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–1780.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601, Seattle, Washington, USA. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of 24th International Conference on Computational Linguistics (COLING 2012): Posters*, pages 549–558, Mumbai, India. The COLING 2012 Organizing Committee.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 114–123, Sofia, Bulgaria. Association for Computational Linguistics.

Diederik P Kingma and Lei Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference of Learning Representations (ICLR)*.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK. Association for Computational Linguistics.

R. Leaman, Rezarta Islamaj Dogan, and Z. Lu. 2013. Dnorm: Disease name normalization with pairwise learning to rank. *Bioinformatics*, 29:2909–2917.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732. Association for Computational Linguistics.

Nut Limsopatham and Nigel Collier. 2015. Adapting phrase-based machine translation to normalise medical terms in social media messages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1675–1680, Lisbon, Portugal. Association for Computational Linguistics.

Nut Limsopatham and Nigel Collier. 2016. Normalising medical concepts in social media texts by learning semantic representation. In *Proceedings of the Association of Computational Linguistics Annual Meeting*, pages 1014–1023, Berlin, Germany.

Zhiyong Lu, Hung-Yu Kao, Chih-Hsuan Wei, Min-lie Huang, Jingchen Liu, Cheng-Ju Kuo, Chun-Nan Hsu, Richard Tzong-Han Tsai, Hong-Jie Dai, Naoaki Okazaki, Han-Cheol Cho, Martin Gerner, Illes Solt, Shashank Agarwal, Feifan Liu, Dina Vishnyakova, Patrick Ruch, Martin Romacker, Fabio Rinaldi, Sanmitra Bhattacharya, Padmini Srinivasan, Hongfang Liu, Manabu Torii, Sergio Matos, David Campos, Karin Verspoor, Kevin M. Livingston, and W. John Wilbur. 2011. The gene normalization task in BioCreative III. *BMC Bioinformatics*, 12(8):S2.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 388–395, Arlington, Virginia, United States. AUAI Press.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web*, pages 101–110. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the International Conference on Learning Representations (ICLR)*.

George Miller. 1998. WordNet: An Electronic Lexical Database. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha,Quatar. Association for Computational Linguistics.

Karen O'Connor, Pranoti Pimpalkhute, Azadeh Nikfarjam, Rachel Ginn, Karen L Smith, and Graciela Gonzalez. 2014. Pharmacovigilance on Twitter? mining tweets for adverse drug reactions. In *AMIA annual symposium proceedings*, volume 2014, page 924. American Medical Informatics Association.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a seamless integration of word senses into downstream nlp applications. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1857–1869, Vancouver, Canada. Association for Computational Linguistics.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

L. Rimell, J. Maillard, T. Polajnar, and S. Clark. 2016. RELPRON: a relative clause evaluation dataset for compositional distributional semantics. *Computational Linguistics*.

H. Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24:97–123.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Peter D Turney. 2014. Semantic composition and decomposition: From recognition to generation. *arXiv preprint arXiv:1405.7908*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference of Learning Representations (ICLR)*.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234, New York, NY, USA. ACM.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601.

Zhigang Wang and Juanzi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint*

Conference on Artificial Intelligence, IJCAI'16, pages 1293–1299. AAAI Press.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Transg : A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2316–2325, Berlin, Germany. Association for Computational Linguistics.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2659–2665. AAAI Press.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International Conference of Machine Learning (ICLR)*, pages 40–48.