Exploring the Combination of Contextual Word Embeddings and Knowledge Graph Embeddings

Preprint · April 2020

CITATIONS
0

READS
21

4 authors, including:

Léa Dieudonat
University of Lorraine
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Phyllicia Leavitt
University of Lorraine
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Esteban Marquer
University of Lorraine
1 PUBLICATION **0** CITATIONS
SEE PROFILE

EXPLORING THE COMBINATION OF CONTEXTUAL WORD EMBEDDINGS AND KNOWLEDGE GRAPH EMBEDDINGS

A PREPRINT

Lea Dieudonat*
IDMC, Université de Lorraine
Nancy, France

Kelvin Han*
IDMC, Université de Lorraine
Nancy, France

Phyllicia Leavitt*
IDMC, Université de Lorraine
Nancy, France

Esteban Marquer*
IDMC, Université de Lorraine
Nancy, France

April 20, 2020

ABSTRACT

"Classical" word embeddings, such as Word2Vec, have been shown to capture the semantics of words based on their distributional properties. However, their ability to represent the different meanings that a word may have is limited. Such approaches also do not explicitly encode relations between entities, as denoted by words. Embeddings of knowledge bases (KB) capture the explicit relations between entities denoted by words, but are not able to directly capture the syntagmatic properties of these words. To our knowledge, recent research have focused on representation learning that augment the strengths of one with the other. In this work, we begin exploring another approach using contextual and KB embeddings jointly at the same level and propose two tasks – an entity typing and a relation typing task – that evaluate the performance of contextual and KB embeddings. We also evaluated a concatenated model of contextual and KB embeddings with these two tasks, and obtain conclusive results on the first task. We hope our work may contribute as a basis for models and datasets that develop in the direction of this approach.

Keywords contextual word embeddings · knowledge graph embeddings · ELMo · ComplEx · entity type prediction · entity relation prediction · FreeBase · FreeBase-NewYorkTimes

1 Introduction

Many methods in Natural Language Processing (NLP) rely on embeddings to process text and knowledge, particularly methods involving the use of deep learning. The quality of embeddings have been shown to significantly improve the results of a wide range of downstream tasks, and a lot of effort have been made towards the development and training of embedding models during the past two decades.

There are two major trends in the literature on embeddings currently: textual embeddings [35, 6, 28, 30, 8, 31, 1, 5, 23, 18, 16] and more recently knowledge embeddings [36, 19, 33, 15, 37, 41, 4, 20]. Also, in the last three years, attempts have been made to augment textual and contextual embeddings with knowledge [22, 29, 42], or knowledge embeddings with textual information [24, 27]. However, most of these works are recent and focus on augmenting either of the two types of embeddings with additional information from the other. In this manner, there is one kind of information (textual or knowledge) which stays at the core of the model and serves as the input in practice, while the other is used to improve the internal representation of the model. To our knowledge, no results has been published of approaches embedding knowledge and contextual information together, using both as the input to the model (instead of augmenting

*By alphabetical order of authors' last names, all authors are first authors.

one with the other), and no evaluation method has been defined that is comparable in the same way for both knowledge and textual embeddings.

We aim to address this gap in the literature by proposing a set of tasks to evaluate the performance of embedding models, designed in a way that knowledge and textual information can be used on the same level. Such evaluation methods are important towards the development of embedding models using both types of information as input.

First of all we will provide an overview of the existing embedding methods and the techniques for evaluating them (section 2, page 2). We will then detail how we approach the problem (section 3, page 7) and how we adapt existing datasets to our needs (section 4, page 8). Finally, we will describe our experimental setup (section 5, page 11), the results (section 6, page 13) we obtain and potential future work (section 7, page 13).

2 Related work

In this section we outline the major trends in the literature on embedding models for text and knowledge bases. We discuss some techniques that have been developed to combine these resources. Finally, we elaborate on the way to evaluate such models.

2.1 Embedding text

Machine Learning methods are widely used in NLP problems such as question answering, search engines, and human-machine interaction. Because Machine Learning models require numerical input, it is necessary to transpose textual data into numerical values when treating NLP problems. Given a vocabulary of a language, a possible approach is to assign singular numerical values, or to index the vocabulary. This very simple solution would not, however provide information about the similarity, relatedness, and distributional properties of words [28]. One approach to capturing this information is through the use of word embeddings.

2.1.1 "Static" word embeddings

Word embeddings – which may also be called vector-space models or distributional semantics models – are the representation of words as vectors. Distributional semantics corresponds to a concept in linguistics first introduced by Wittgenstein in his work, Philosophical Investigations, arguing that a word may be represented by the way in which it is used [38]. This, along with the idea that language has a distributional structure [13], has been the driving factor behind the development of word embeddings, which are derived from a probability distribution of words found in their context, obtained by training on large, unlabeled textual corpus. The approaches to training word embeddings may be broken into three broad categories: pre-neural, neural and contextualized.

In the classical approach, pre-neural models output word embedding by applying Latent Semantic Analysis (LSA) to a term-document matrix, making use of matrix dimension reduction by means of Singular Value Decomposition (SVD), which is a matrix decomposition operation.

Recently, the development of neural techniques for training word embeddings has made an impact on the NLP community. Namely, the model developed by [16] made available embeddings pre-trained on large corpora by means of neural network techniques. While there are different techniques for encoding them, traditional, "static" embeddings such as Word2Vec, Glove and fastText represent words by a single vector per token of a vocabulary.

Although these embeddings encode what we have previously referred to as "distributional semantics", they do not capture per se, information such as hyponymy, synonymy, and other linguistic and/or factual information which is present in linguistic resources [24].

2.1.2 Contextual word embeddings

Contextual word representations was first introduced by [23], and allows the encoding of embeddings which vary "dynamically" with the input sequence. For example, if given a sentence "John and his friends play soccer", and "I attended the play", there would be different output for the polysemous token "play". Such embeddings are achieved by training language models and outputting not only the word embeddings themselves, but also the weights that are achieved through training[28]. Significant improvements have been observed on NLP tasks which use contextualized embeddings as input [23].

Since the introduction of ELMo in 2018, other contextual embedding models have been developed. While they differ in the neural network architectures and the language modeling tasks they are trained on, all these models have in common the ability to represent words in context. What is actually learned in each of these embedding models is however an

open topic of research[8]. Although [23] observe that the word vectors generated from the first Bi-LSTM[2] layer seem to better capture syntax and the second layer seem to capture semantics better.

2.2 Knowledge graph embeddings

KBs are a store of information for use by computer systems. KBs with structured, relational information are typically referred to as knowledge graphs (KGs). They contain relational information between entities, as well as the properties of these entities and their relations. Thanks to the relative scalability afforded by crowd-sourcing, very large KGs have been built. These include Freebase (see section 4 page 8) and Google's Knowledge Graph, which typically store information about entities and the relations between them as triple with the following form `<subject, relation, object>`. Typically, KB entities include nominals as well as abstract concepts, whereas KB relations may have one or more of the following properties: symmetry, reflexivity, transitivity and equivalence.

However, because of the symbolic nature of their representation and their sizes, KGs are difficult to manipulate[36]. Within the last decade, new approaches have been proposed to represent the discrete symbolic information in KGs in continuous vector spaces, by embedding entities and relations in these spaces. Two broad classes of such KG embeddings approaches have emerged - namely, translation-based models (subsubsection 2.2.1) and latent feature models (subsubsection 2.2.2).

2.2.1 Translation models

The `TransE` model proposed in [4] is the earliest member of the translational class of KG embedding models. It works by modelling the relation between a head entity and a tail entity as a translation in vector space, and embeds KG information by optimising the positions of entities and relations globally across the embedding space. Figure 1 (page 4) includes a depiction of how entities and relations are represented in the TransE model.

The TransE model was however found to be too restrictive in representing relations. While it is suitable for embedding KGs with one-to-one relations, it is unable to sufficiently embed the full properties of KGs that contain one-to-many, many-to-one and many-to-many relations [36, 37]. Models extending the capabilities of TransE have been proposed, these include `TransH` [37] and `TransR` [15], as well as more recent models which seek to extend specific aspects of each of these three models.

TransH, proposed in [37] casts the representation of each relation as a hyperplane in the vector space. Relations between entities are then translations across these hyperplanes. In doing so, TransH expands the capability of TransE to represent multiple relations between entities. An illustration of the approach taken in TransH can be found in Figure 1 (page 4).

The TransR model in [15] adopts a different approach by embedding relations and entities in separate vector spaces. Specifically, each relation in TransR is represented by a projection matrix. An application of the projection matrix of relation r on the entities embedding space will project the entities onto r's embedding space. Therefore, a translation of r on the head entity in this r-projected space will identify tail entities it is related to by r, within a neighbourhood of the translation. In doing so, TransE extends the representational power of TransH by allow translations beyond hyperplanes, in higher dimensional vector spaces.

While these extensions increase the expressivity of the initial TransE model, the expressivity comes at the expense of higher complexity [36]. For instance, TransR requires a projection matrix for each relation in the KG, thereby increasing significantly the training and memory requirement for the model.

2.2.2 Latent feature models

Another class of KG embedding models approaches the problem by representing KG information in the form of a tensor[3]. The KG tensor is then factorised to obtain two component tensors, one representing entities and another representing relations [19]. These tensors capture the latent semantic properties of the entities and the relations respectively. The interaction (via tensor dot product operations) between the entity, relation and entity-transposed tensors provides an indication of the plausability of a relation between two entities.

The `RESCAL` model by Nickel et al [20] is representative of this approach and is depicted graphically in Figure 2 (page 4). RESCAL's tensor factorisation approach has, however, a higher complexity compared to TransE and TransH[36]. It has

[2]ELMo's architecture comprises a three-layer structure where layer zero is the character-based context independent layer, followed by two Bi-directional Long-Short Term Memory (Bi-LSTM) layers.

[3]Such KG tensors tend to be sparse due to the nature of the entities and relations that are found in large KGs.

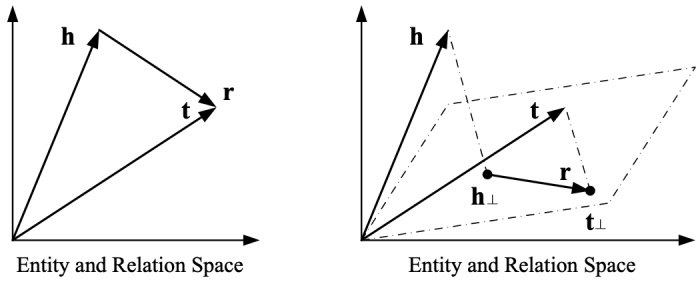

Figure 1: Visualisations of translational KG embedding models in lower-dimensional space. The figure on the left illustrates the TransE approach to embedding entities and relations, the figure on the right illustrates the same for the TransH model. Source: [36]

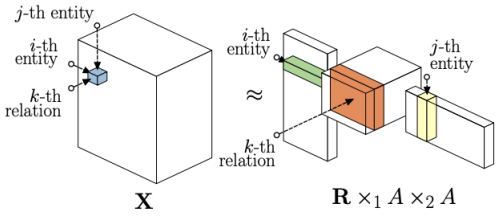

Figure 2: Factorization of an adjacency tensor X (left) using the RESCAL model. The interaction, via dot product, of the factorised A matrix (containing the latent information about entities) and **R** matrix (in orange on the right) as well as the transposed A matrix, provide information about the relation between pairs of entities. Source: [20]

a space complexity of $\mathcal{O}(nd + md^2)$ and a time complexity of $\mathcal{O}(d^2)$, compared to a space complexity of $\mathcal{O}(nd + md)$ and a time complexity of $\mathcal{O}(d)$ for TransE and TransH[4].

Another latent feature models, `DistMult` [41] addresses the complexity issue faced by RESCAL by restricting the matrix, that represents relations, to be a diagonal matrix [36]. This reduces the number of parameters for representing each relation to $\mathcal{O}(d)$. While DistMult addresses the complexity issue faced by RESCAL, and the representation approach taken by both of them *"scale well and can naturally handle both symmetry and (ir-)reflexivity of relations; using an appropriate loss function even enables transitivity"*[33], both models will require a substantial increase in the dimension of the matrices in order to represent asymmetric relations.

The `ComplEx` [33] model proposed by Trouillon et al (2016) extends the expressivity of DistMult by leveraging the properties of complex numbers, where the result of a dot product of two complex-valued matrix is asymmetrical[5]. This property allows ComplEx to represent asymmetric relations with the same level of complexity as DistMult.

2.3 Augmenting word embeddings with knowledge graph embeddings

There are many ways that experimenters have tried to augment embeddings with information encoded from lexical resources. Because many lexical resources are available in the form of an ontology [17, 2], a portion of these approaches have been geared towards learning an alignment of word embeddings to knowledge graph embeddings [5].

There are two broad approaches concerning the alignment of "static" word embeddings with KG embeddings. Firstly, *"joint specialization models modify the learning objective of the original distributional model by integrating the constraints into it"*. [9] for example, will modify the learning objective in order to learn the distribution of the entity-to-entity co-occurrence (rather than co-occurrence of a word in its textual context), whenever possible. On the other hand, post-processing models will carry out what is commonly called retro-fitting [12, 35]. Retro-fitting involves fine-tuning pre-trained word embeddings by presenting constraints. For example, in the case of retro-fitting word embeddings with KB information, such constraints introduced would be similarity constraints between two particular entities. There have

[4] Where n and m denote the number of entities and relations in the KG being embedded, and d denote the dimensionality of the entity embedding space.

[5] $\mathtt{A} \cdot \mathtt{B} \neq \mathtt{B} \cdot \mathtt{A}$, where `A` and `B` are complex-valued matrices.

been many experiments which take this approach, namely [10], who develops a generalized graph-based method to integrate lexical resources. This method iteratively fine-tunes word embeddings, altering the euclidean distance between vectors to better respect related and similar words provided by the WordNet FrameNet and paraphrase database (PPDB) KBs.

The current existing joint contextualized embeddings and KG embeddings models have been achieved by implementation of an attention mechanism [29, 22]. As attention mechanisms and the methods in these experiments are specific to the architecture of BERT embeddings, these techniques could not be applied as such to ELMo's contextualized embeddings. To our knowledge, there are no existing models which could be used to augment ELMo embeddings with knowledge base information.

2.4 Evaluating embedding models

There exists two main approaches to embedding model evaluation. The first family of methods relies on tasks that we try to achieve using the embeddings and then score the performance on the tasks. With the evaluation being done not at the level of the embedding model but later in the processing, such methods are usually called *extrinsic evaluation methods*. They are sometimes referred to as *downstream tasks* as such a task is used to evaluate the model.

In contrast, the second family of methods does not rely on external tasks and are thus called *intrinsic evaluation methods*. They typically rely on metrics applied directly to the embeddings or the embedding space, so at the level of the embedding model.

Both intrinsic and extrinsic evaluations of an embedding model can be interpreted in terms of linguistic properties of language the model should capture (like similarity and relatedness, see [26]). One should note however that it is only an interpretation of the results and in the case of downstream tasks, of the way the task is designed.

2.4.1 Intrinsic evaluation

According to [1], *"methods of intrinsic evaluation are experiments in which word embeddings are compared with human judgments on words relations"* (in the case of word embeddings). They categorize intrinsic evaluation methods using human judgments in 2 major categories:

1. absolute evaluation, where automatic metrics applied on human annotated data;
2. comparative evaluation, in which humans evaluate the model by comparing the outputs of multiple models, such method typically relying on crowd-sourcing.

They further categorize those methods based on how the human assessment is collected, namely conscious evaluation (with time to think about the decisions) and unconscious evaluation (without time to think about the decisions or with the evaluation being peripheral to what the evaluator is asked to do).

They mention that this typology is not exhaustive as it does not account for methods that do not rely on human judgment, like those comparing the results of textual embedding to a reference KB or those based on intrinsic properties of language. In a more general view, this last category includes the methods using the structure of the ~~data~~ training data (text or KB) to evaluate the embedding models.

Here is a list of the main intrinsic evaluation methods present in [1, 11, 31].

Semantic similarity-based methods assume that the distance of the elements in the embedding space represents their similarity. Closer elements would be more similar. Typically, the distance measure used is the cosine distance. A wide range of datasets of pairs of words with human similarity scores is available, as it is the most widespread intrinsic evaluation method for embeddings. This method is a straightforward evaluation of the modeling of similarity. However as explained in [11] this kind of method is biased by what the annotators understand by semantic similarity. This method is a straightforward evaluation of the modeling of similarity.

Synonymy-based methods are based on the detection from a set of elements the closest to a given element. This kind of method is very close to semantic similarity-based methods as it is based on the same assumptions and can be interpreted in the same way. Synonymy-based methods are more clearly linked to similarity however, as they explicitly evaluate synonymy.

Analogy-based methods are based on the idea that if the embeddings correctly encode the attributes of the elements encoded, we should be able to execute basic arithmetic using the embeddings. A stereotypical example is that of *king* - *man* + *woman* = *queen*, with *queen* being to *woman* what *king* is to *man*. A major drawback of this family of methods is the lack of precise evaluation metric. We can consider this method as an evaluation of the ability to model relatedness.

Thematic fit-based methods evaluate the ability to determine semantic roles of a predicate, in other words, the ability of the model to propose the most relevant noun to use with a given verb and a given role. It can be seen as a measure of the similarity or of the ability of the model to learn to categorize the elements by their semantic role.

Semantic (embedding) space coherence-based methods try to evaluate the quality of the neighborhood of elements in the embedding space. This can be interpreted as an evaluation of the internal coherence of the embedding space. As described in [11], a typical task to propose to the human evaluator would be to select an element and its 2 nearest neighbors in the embedding space as well as a randomly chosen element. The evaluator should be able to determine that the randomly chosen one is an outlier, an "anomaly".

Linguistic features alignment methods described in [11] evaluate how well the dimensions of a word embedding can be aligned with dimensions of so-called linguistic vectors computed from a semantically annotated corpus. Because we rely on a semantically annotated corpus we do not need direct human annotation or rating for this task. The results of this evaluation method have been shown to be fairly correlated to performance in downstream tasks, making it a good performance estimate. Also, by comparing the embedding with linguistically justified dimensions, this method focuses on how easily we can interpret the embedding themselves according to linguistically meaningful dimensions.

Clustering methods are methods for which "the task is to split [a set of words] into subsets of words belonging to different categories" ([1]) and the evaluation of the quality of the clustering by comparing it to the known categorisation. "Possible criticism of such method could address the question of either choosing the most appropriate clustering algorithm or choosing the most adequate metric for evaluating clustering quality." ([1]) Also, in our opinion, such methods are more extrinsic than intrinsic because they rely on a additional model to solve a task. Indeed, from the own words of [1], clustering evaluation methods are based on a task. Such method can be interpreted as a measure of whether the model learns to categorize the elements in a similar manner as the reference categorization.

Outlier detection methods also rely on a task which is to identify the element which is different from the rest within a provided set of elements, based on a known categorisation. Those methods are quite similar to clustering methods in that they deal with element categories, and can thus be interpreted in a similar manner and are, in our opinion, more extrinsic than intrinsic methods.

As mentioned in [1], *"NLP engineers who are more interested in dealing with downstream tasks (for instance, semantic role labeling) usually evaluate the performance of embeddings on such tasks, while computational linguists exploring the nature of semantics end to investigate word embeddings through experimental methods from cognitive sciences."* This insight can serve as an hint to why embeddings-related literature focuses on downstream tasks, even if intrinsic evaluation allows for a more targeted analysis of the inner workings of an embedding model. It is indeed a field where the practical use of an embedding model is a core factor of whether the model will be used or not.

2.4.2 Extrinsic evaluation

In practice, extrinsic evaluation of embedding models involves a task or set of tasks, the choice of which should be guided by the properties of the embedding model that we want to evaluate. Such a task is usually defined by:

1. the goal of the task (ex: detecting similar words, predicting relations, *etc.*);
2. a machine learning model separate from the embedding models, that will use the embeddings as features to solve the task; as mentioned by [1], such a model is typically supervised;
3. data on which the embedding models will be applied to obtain the embeddings and perform the task;
4. a way to evaluate the performance on the task.

A wide variety of downstream tasks have been designed over the years to evaluate embedding models, the main ones described in [1] and [36] respectively for word and KB embedding models. Most methods used for word embedding models are applicable as-is on contextual embeddings, though with slightly different data ([32]).

While the tasks are quite similar between textual (word or context) and KB embedding model evaluations, the kind of data used as input for the embedding model is quite different. For textual embeddings the data is text, while for KB embeddings the data is either triples or KG node identifiers.

We will describe the main families of tasks used and how they are designed for both textual and KB embeddings. For an overview of the datasets available for each task, see the corresponding sections in either [1] or [36].

Relation (or link) prediction is a type of task focused on identifying the relationship between two elements. While very used for KB embeddings, [22] presents a contextual embedding version.

To perform this task with KB embeddings, the data used is a triple `<subject, _, object>` in which the relation has been removed and serves as the target of the prediction task. For textual embeddings, we have something of a textual adaptation of the KB version of the task, with the object and subject of the relation lexicalized – put into a textual form, either a word or a word in context. There is a variant of textual relation prediction called *relation extraction*, in which the relation to predict is described in the text (ex: "Barrack Obama was elected president of the USA", the relation of being the president of is present in the text) making the task more of an extraction of the relation from the text than a simple prediction of the relation between entities.

While usually what is predicted is the relation between the elements, the relation can be provided with the source or the target of the relation, the remaining one being the prediction target. In that case, it can be called *entity prediction*. Relation and entity prediction are more generally called *triple completion* as they amount to filling the missing part of a triple.

Entity typing (or classification) is the task of predicting the type of an entity represented by a KG node or a word. Entity typing is typically used for KB embeddings as type information is usually included in the knowledge base, but [22] presents a contextual embeddings version. This family of tasks can be interpreted as a measure of whether the model learns to categorize the elements by their nature.

With KB embeddings we embed a single entity and try to determine its type. It is usually quite easy to acquire the gold labels for this task with KG, as typically every node is attributed a class. With textual embeddings we take a word (and its context if necessary) and predict its type in a similar manner. It is however a bit less straightforward to determine the type of the entity in this case than with KBs.

Entity resolution is the task of checking if two words or KG nodes represent the same entity. This kind of task is supposed to evaluate the ability to model semantic similarity in a rather straightforward manner. For textual embeddings in particular, the task can be used to evaluate how synonymy is encoded in the model.

For KB embeddings we embed pairs KG nodes known to represent or not the same object, while in the case of word embeddings, we embed pairs of words.

There is also variant of the task called *word in context* (WiC) [22] which corresponds to the contextual embedding version of the task, for which we determine if words used in their respective contexts are used with the same sense.

Textual entailment detection is a task for which we determine if a chunk of text logically entails another. A typical example would be "Mary gave John a book" that implies that "Mary gave a book". This task should evaluate the ability of the model to encode logical relations, and fits more contextual embeddings and so-called sentence embeddings than word embeddings.

Triple (fact) classification is a task for which we determine if a triple describes a true information or not. Reusing the example from [36], the triple <Alfred Hitchcock, Director Of, Psycho> is true while <James Cameron, Director Of, Psycho> is false.

Those generic task families are the most used because they have a wide variety of datasets available and are quite simple to set up. Also, while those relatively simple tasks do not evaluate performance for a target application, they can be used to estimate the performance when we rely on properties of the language similar to those evaluated by the task. Such an approach was used in [22], where the authors use relation extraction, word in context and entity typing to evaluate their knowledge enhanced contextual embedding. While their method is close to what we propose, they evaluate their model exclusively on textual data.

Tasks which are more closely related to real world use cases are also used, for example triple extraction from text, question answering, recommender systems, Named Entity Recognition (NER) or semantic role labelling. Those tasks are harder to analyse to linguistically interpret the performance of the model, but they provide a better insight on how the embedding model impacts performance for a target application.

3 Approach

In the following section we develop the approach taken in establishing a framework of comparative analysis between the use of contextualized embeddings and KG embeddings as input of a Machine Learning NLP problem. We first define the two tasks that we have

Model	MRR	Hits@10
TransE	0.221	0.641
DistMult	0.242	0.824
ComplEx	0.242	0.840

Table 1: Comparison of the performance of various KG Embedding approaches on the FB15k dataset. Source: [33]

3.1 Task 1: entity typing

Given a text with the label available for the entity and the relation between them, the aim of the entity type prediction is to detect and predict semantic types of a named entity in knowledge graphs. As entities can have multiple types, we are predicting the set of types an entity should have. It is beneficial for a large number of NLP tasks such as entity linking, relation extraction, question answering and knowledge base population. Linking this task to our previous hypothesis, entity typing does not directly evaluate similarity nor relatedness, but something like meaningfulness of the representation.

3.2 Task 2: relation prediction

Relation prediction is a task that take two entities and predict the relations between these two elements. We have the sentences that contains information on the labels of the entities, their location in the sentence, but not the relation between the entities. As this task is directly linked with the concept of relatedness, it allows us to verify our previous hypothesis.

3.3 Chosen frameworks

We chose to conduct our experiments with ELMo. It has a bidirectional language modelling architecture that is based on a predict neighbouring-word task. Compared to other contextual embedding models such as BErT[6], ELMo has a relatively less complex architecture and its training task is the same as that used in classical "static" embedding models such as Word2Vec, albeit ELMo is trained on the next as well as the previous world. The neighbouring word language modelling task is well-studied, and allowed us to better explain the results of our experiments. We used the large version of the official ELMo pre-trained model[7], where each contextualised word is represented by a 512-dimension vector.

For KG embeddings, we chose to work with ComplEx because it has been shown to perform well on KGs. On the FB15k dataset, ComplEx was shown to perform better compared to TransE and DistMult on the prediction of relations in the FB15k dataset (see Table 1, page 8). To obtain the ComplEx embeddings, we used the PyTorch-BigGraph[8] implementation of ComplEx [14], trained on FB15k (see subsection 4.1 page 9), a subset of Freebase. The resulting ComplEx embedding model represents entities as 400-dimension vectors.

3.4 Hypothesis

Starting with the assumptions that KB embeddings represent entities in triples and word embeddings represent entities from unlabeled and unstructured text, our initial hypothesis is that KB embeddings are most likely to obtain significant scores in entity typing and relation prediction, when compared to with word embeddings. According to the structure of the triples, entities are easier to spot in KB embeddings. Our second hypothesis should rely on the fact that a combined model that join the KB embeddings and the contextual embeddings can perform at least equally or better at most than the maximum possible score for one model on one task. Another point to be mentioned is the similarity dimension where contextual embeddings are supposed to perform better than KB embeddings. This point will not be exploited in the next steps of our work.

4 Data

The overall aims of our experiments being to compare the efficacy of contextualized embeddings with knowledge graph embeddings, this factored into the kind of data required for experiments. As previously mentioned, in order to access

[6] BErT utilises multi-head attention, and its training tasks include token masking and next sentence prediction.

[7] `https://allennlp.org/elmo`

[8] `https://github.com/facebookresearch/PyTorch-BigGraph`

ELMo embeddings of a given word, we are required to feed a full sentence in which it is situated. For this particular reason, both experiments necessitated not only the lexicalization of the entities themselves, but of these entities in their context. On one hand, for the task of entity typing, we required a sentence in which situates each lexicalized entity. On the other hand, for the task of relation prediction, we would need a dataset containing lexicalizations of triples, not just the entities themselves. In order to carry out experiments for both of our tasks with consistency, we sought to use datasets and resources which are based on the same KB. We found the Freebase KB to be the most appropriate to fit these needs.

Freebase is an open collaborative KB launched in 2007 by Metadata and bought by Google in 2010. The project was brought to a close, as Google migrated the data to Wikidata in 2015 in order to support this fast growing effort to lead an open collaborative KB [21]. The motivation to basing our experiments off of this KB are as follows:

- an existing subset of this KB, Freebase 15K [4], makes experiments more manageable given our computational power constraints;
- there are existing extensions of the Freebase 15K which are appropriate for the typing task [39];
- there are available lexicalizations of Freebase triples based off distant supervision [25].

4.1 Freebase15K

Freebase 15K (FB15K) is a subset of the full Freebase KB, which has been extracted and pre-processed by [4] in order to carry out preliminary experiments. In order to produce the subset FB15K, experimenters targeted entities and relations which are also present i the still existing Wikidata [34] database while maintaining at least 100 mentions in the larger Freebase. It has been widely used for experiments in both embedding knowledge graph embeddings [37, 15, 39] as well as in entity-focused tasks such as entity typing [40].

The FB15K dataset contains entity codes, the lexicalisations of these entities, their types, relations, and the triples split into train, valid, and test datasets. Counts for the entities and relations present in the FB15K can be seen in Table 2 (page 9).

	Count
Entities	14,951
Relationships	1345
Types	4054
Train set	483,142
Valid set	50,000
Test set	59,071

Table 2: Elements and triple counts in FB15K

	Count
Named entities	13533
Non named entities	1408

Table 3: Counts of named entities in FB15K

	Count
Train	11607
Test	1268

Table 4: FB15K sample counts after filtering

As previously mentioned, we not only needed a manageable sized KB to work from for our tasks, but we also necessitated on one hand, sentences in which a given entity is situated, and on the other, surface realizations of triples (discussed in subsection 4.2). For this reason, along with the FB15K dataset, we have made use of an extension of this dataset containing descriptions of the entities which was extracted by experimenters [39]. In order to process this dataset, we followed the following procedure:

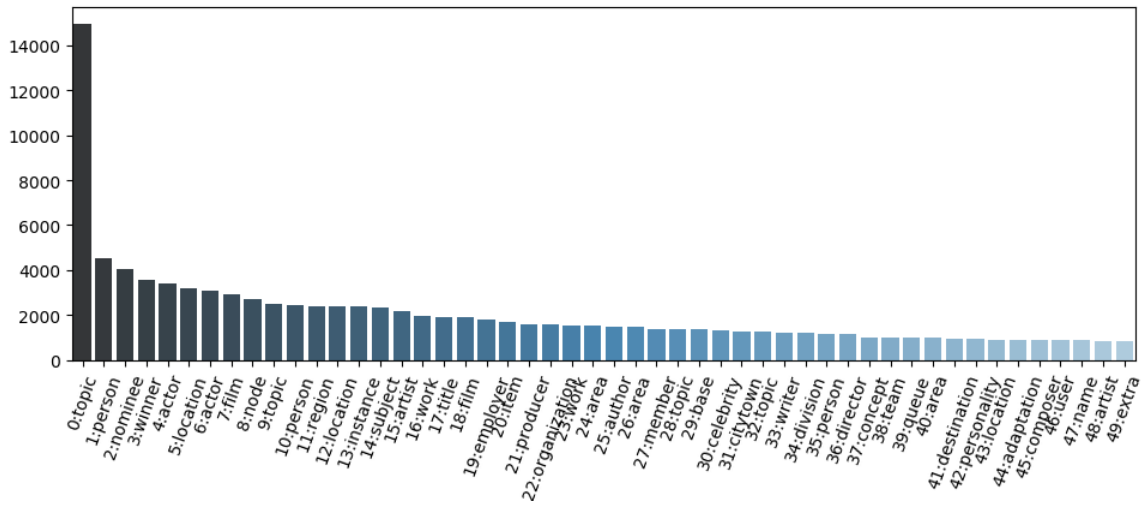

Figure 3: Distribution of entity types

1. look up and tokenize the full lexicalization of the entity code;
2. select entity head-word based on most frequent token across description;
3. search for mention in its corresponding description;
4. drop items whose mentions do not match the lexicalization (with allowance of up to 1 token, as in a middle name, which separates original tokenized lexicalization);
5. select candidate sentences with mentions matching this criteria;
6. take first sentence from candidate list, using headword and it's index in the sentence as position in context.

Examples 1 and 2 show instances where an entity description did not pass through our pre-processing would be.

(1) ACTOR-GB: An actor is a person portraying a character in a dramatic or comic production; she or he performs in: film, television, theatre, or radio *etc.*

(2) EMILY BLUNT: Emily Olivia Leah Blunt is an English actress. She has appeared in The Devil Wears Prada, The Young Victoria, The Adjustment Bureau, and Looper. She has been nominated *etc.*

The previous examples did not carry over for the reasons that the entity mention (appearance in the text) did not match that of our input data, or there were more than one tokens interrupting the succession of tokens which compose the entity lexicalization. In example 3 our processing took hold of the entity. In the example the selected head-token is underlined, the full lexicalization in bold, and the context sentence in italics.

(3) E1 MUSIC: ***E1 Music**, the primary subsidiary of Entertainment One LP, is an independent record label in the United States.* It is widely regarded as the most successful independent record label in the United States, having garnered the most Billboard hits of any independently-owned music label in history *etc.*

4.2 Freebase-NewYorkTimes

The Freebase-NewYorkTimes dataset ("FB-NYT") contains surface realisations of entities and relations found in Freebase [25]. The surface realisations are from the New York Times annotated corpus[9] released by the Linguistics Data Consortium, which is made up of 1.8 million news articles that were published by the American newspaper, The New York Times, in a 20-year period between 1987 and 2007.

[9] `https://catalog.ldc.upenn.edu/LDC2008T19`

	Before filtering	After filtering
Number of sentences	66,196	13,874
Number of entities	14,664	798
Number of relations	24	16
Number of samples	111,327	29,492

Table 5: Composition of FB-NYT dataset before and after our filtering

The dataset was built automatically[10] on the following basis[11]: "If two entities participate in a relation [as part of a triple in a KG], at least one sentence [in a text corpora] that mentions these two entities might express that relation." ([25]) Although the dataset was automatically created, manual evaluation of a sampling of the generated FB-NYT with two human annotators indicates, with statistical significance, that the approach is able to identify relation instances with a precision of 91%.

FB-NYT is suitable for our relation type prediction task as it contains the surface realisation of triples found in Freebase. The surface realisation provides us with the context that allows us to obtain the contextual word embeddings for the entities. However, our experiments are conducted with knowledge graph embeddings trained on a subset of Freebase, FB15k. As such, we filtered FB-NYT for relations and entities that were found in FB15k. Table 5 (page 11) provides an overview of the sentences, entities, relations and samples before and after our filtering.

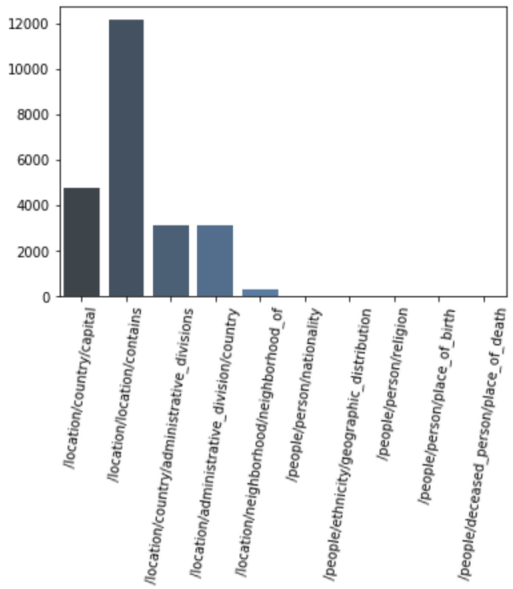

Figure 4: Distribution of relation types in the training set of our filtered FB-NYT corpus

5 Experiments

We present in this section the main aspects of our experimental setup: how do we explore the possibility of combining knowledge and text in the embedding, how do we perform the tasks and how we determine the performance of the models on the tasks.

[10]This was done with the following steps (i) named entity recognition, on all of the sentences in the New York Times annotated corpus, (ii) string match of the entities mentions found with the canonical form of the entity name in Freebase, (iii) feature extraction of the entities matched; and (iv) factor graph and constraint-driven semi-supervision to jointly decide if a relation exists between two entities as well as what the relation is.

[11]This is a relaxed version of the original *distant supervision assumption* that "if two entities participate in a relation, all sentences that mention these two entities express that relation."

5.1 Proof-of-concept combined model

For our experiments we introduce a simple embedding model combining KB and contextual information in the same embedding.

The model simply concatenates the embedding generated with the two pre-trained models we use for KB and contextual data (BigGraph and ELMo respectively). The resulting embedding is almost twice as large as each of the separate embeddings (512 cells from ELMo and 400 from BigGraph so 912 cells once concatenated). However, it contains the information of both the KG node representation and the textual description exactly in the same way as ELMo and BigGraph do separately, without removing or adding anything. As such, this model is likely to be only at the surface of what can be gained by embedding text and knowledge together.

This model is meant as a proof-of-concept to study whether embedding KB and contextual information at the same level can benefit the downstream tasks. Also, if such a model make a significant improvement of the performance relative to a KB or contextual representation alone, this result would encourage the enrichment of KB and textual datasets with textual and KB data respectively. Lastly, as the model we use is extremely simple, it can serve as a baseline for more complex models relying on both kinds of data.

5.2 Task setup

The tasks we have chosen to develop and evaluate – namely entity typing and and the relation prediction tasks (using the corpora described respectively in subsection 4.1, page 9 and subsection 4.2, page 10) – are both multi-class classification tasks. Indeed, for typing we classify an entity embedding into a set of types, and for relation prediction we classify a pair of entity embeddings into the relation between the entities. A slight difference between the tasks is that we have only one relation to predict per sample (multi-label single-output setting) while we have a varying number of types per sample (multi-label multi-output setting). We thus use a very similar approach in both cases:

1. we generate the embedding(s) for the entity or (entities) to classify;
2. we train a Logistic Regression classifier on the embeddings from the training set;
3. we evaluate the quality of the prediction using the metrics defined in subsection 5.3 (page 12).

Our intent with the tasks is not to obtain a high performance on the task but to see how the performance is impacted by the embedding model. A simple classifier like the Logistic Regression allows us to shift the performance more on the quality of the embedding. The logistic model may, however, not be complex enough to handle the large dimensionality of the embeddings.

Also, to maintain an "all other things being equal" setting when building the task, by providing the task module with the same "blind" embedding model structure taking both the KB and contextual information as input whether the model uses the information to produce the embedding or not. This allows us to have a systematic classification process, with a single code applied on the embeddings regardless of their input and of the embedding size.

5.3 Evaluation metrics

For our experimental tasks, we used `Precision@n`, Mean Average Precision@k (`MAP@k`), and Mean Reciprocal Rank (`MRR`) as evaluation metrics. These are commonly used evaluation metrics in the field of information retrieval, of which KBs are widely used in. These metrics are relevant when there are multiple labels/answers for a given sample.

Precision@n is a measure used for data with samples that contain more than one correct label, i.e. a set of labels. It measures, for a single predicted set of labels, the number of these that are accurate up to a set cut-off position, n. It can be expressed in the following form:

$$\text{Precision@n} = \frac{\text{\# of accurate predictions up to n}}{\text{n}}$$

The MAP@k for a dataset is the mean of the average precision@k (AP@k) for each sample[12]. AP@k is computed as follows:

$$\text{AP@k} = \frac{1}{\text{m}} \sum_{i=1}^{k} \text{P}(label_k), \text{ if the } k^{th} \text{ item is relevant}$$

[12]For a multi-label single output classification task, MAP@k=1 is the same as simple.

where k is a set cut-off position and m is the number of labels for the particular sample.

The MRR for a dataset is the mean of the reciprocal rank (RR) of each sample in the dataset. MRR and RR are expressed in the following form:

$$\text{RR} = \frac{1}{\text{R}}, \qquad \text{MRR} = \frac{1}{\text{N}} \sum_{i=1}^{N} \text{RR}(output_i)$$

where R is the rank of the accurately predicted label and N is the number of samples in the dataset.

6 Results

For the first experiment, the MAP@k=10 and Precision@k=10 scores in Table 6 (page 13) suggest that KG embeddings perform better than contextual word embeddings in an entity typing task. This is in line with our starting hypothesis (see subsection 3.4 page 8). The results also suggest that a representation, that is a concatenation of both models KG and contextual word embedding models, is almost on par with that of KG embeddings.

For the second experiment, in Table 7 (page 13), suggests that KG embeddings perform better than contextual word embeddings in the task of relation typing. Again, this is in line with our starting hypothesis (see subsection 3.4 page 8). Contrary to our initial hypothesis however, the performance of the concatenated representation performs poorer than contextual word embeddings in the same task – the MRR for the former is 1.2 percentage points below that of the latter. One factor for the poorer result of the concatenated representation is that it may carry duplicated information that is found in both the KG and contextual word embeddings. Further work and analysis section 7 (page 13) would be required to validate this hypothesis.

Model	**MAP@k=10**	**Precision@k=10 (mean ± std)**
Contextual Embeddings (1)	0.631	0.449 ± 0.271
KG Embeddings (2)	**0.825**	0.528 ± 0.269
Concatenation (1) + (2)	**0.828**	0.527 ± 0.268

Table 6: Results of the entity typing experiment

Model	**MRR**	**Precision (MAP@k=1)**
Contextual Embeddings (1)	0.750	0.554
KG Embeddings (2)	**0.817**	0.663
Concatenation (1) + (2)	0.738	0.533

Table 7: Results of the relation prediction experiment

7 Future work

The nature of our experiments being of a proof of concept, we have many projected improvements and projects for further research. The majority of our work has been setting up the framework for the two tasks: preparing the data and establishing the evaluation metrics. The areas of improvement on the tasks we have carried out are outlined below:

1. applying PCA on the embeddings in order to test for the "curse of dimensionality" ([3]);
2. using a simple, but higher-capacity, model such as MLP;
3. adapting our model in order to prefer nominals as head words [7];
4. analyse the interpretability of the relation prediction task and the effect of linguistic features such as sentence structure in our setting.

We have made the hypothesis based on our results of the second experiment that, perhaps dimensionality could be an explaination that the scores were lowest for the concatenated input of ELMo embeddings and knowledge graph embeddings. In order to test this hypothesis and further explain the results of our experiments, we propose further

processing the input via principal component analysis in order to reduce the dimensionality, presumably resolving this possible source of error. Although we have privileged interpretability over performance in our experiments, it could be useful to test other, higher-capacity models such as MLP in order to see if our hypothesis holds. Also, the metrics during the evaluation are fine-grained, so an analysis taking into consideration a coarser baseline (such as raw precision) may contribute to a holistic analysis of each of the embedding model's performance.

In addition, we have made some suggestions for further experiments along the line of a linguistic interpretation of our results. Linguistically, we presume that nominals carry information about an entity mention. In fact, in a similar entity-focused experiment setting, [7] has privileged selecting common nouns which are present in multi-word entities in order to capture type information: "Many nominal entity mentions include detailed type information within the mention itself. [Nominal entity mentions] provide a somewhat noisy, but very easy to gather, context-sensitive type signal". We therefor propose prioritizing selection of common nouns during pre-processing for task 1. Finally we propose an analysis of linguistic aspects that may provide explaination for results of our second task.

As previously mentioned in subsection 2.3, there are no currently established methods for jointly embedding ELMo contextualized embeddings with knowledge graph embeddings. While staying within the framework of ELMo and the possible integration of Freebase facts, we propose further methods in terms of feasible ways to study contextualized embeddings. It would be possible to learn a mapping from entities' ELMo embeddings to the corresponding KG embeddings. Mapping embeddings to knowledge embeddings has been previously experimented, as an enquiry into the interpretability of what information "static" embeddings hold [5]. Similar experiments have been achieved by mapping embeddings to definitions. Such an enquiry would be very interesting to pursue [6].

8 Conclusion

In conclusion, we provide two downstream tasks with datasets applicable equally on contextual embeddings, KB embeddings and models using both textual information and information from KBs. Those tasks are typical and generic downstream tasks enabling the estimation of the performance of embedding models on a wide range of applications.

Also, by using contextual and KB embeddings and combining them in a new embedding, we explore a new approach at modelling KB and contextual information on the same level. In particular, with such a simple embedding model we manage to obtain conclusive results for our typing task, demonstrating the interest of using KB and contextual information as input for embedding. While we do not manage to obtain conclusive results for our relation prediction task, we have some hypotheses as to why we have such results and ways to explore them.

There is still a lot to do in this field of trying to bridge the gap between knowledge and text processing, and we hope our work can serve as a basis for models and datasets following the same train of thought.

References

[1] A. Bakarov. A survey of word embeddings evaluation methods. *CoRR*, abs/1801.09536, 2018.

[2] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[3] R. E. Bellman. *Adaptive control processes: a guided tour*. Princeton university press, 1961.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 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., 2013.

[5] J. Camacho-Collados and M. T. Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788, 2018.

[6] T.-Y. Chang and Y.-N. Chen. What does this word mean? explaining contextualized embeddings with natural language definition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6066–6072, 2019.

[7] E. Choi, O. Levy, Y. Choi, and L. Zettlemoyer. Ultra-fine entity typing. *arXiv preprint arXiv:1807.04905*, 2018.

[8] K. Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[9] W. Fang, J. Zhang, D. Wang, Z. Chen, and M. Li. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 260–269, 2016.

[10] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[11] A. Gladkova and A. Drozd. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[12] G. Glavaš and I. Vulić. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45, 2018.

[13] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[14] A. Lerer, L. Y. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich. Pytorch-biggraph: A large-scale graph embedding system. In *SysML, 2019*, volume abs/1903.12287, 2019.

[15] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2181–2187. AAAI Press, 2015.

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

[17] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[18] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *CoRR*, 2015.

[19] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, Jan 2016.

[20] M. Nickel and V. Tresp. Tensor factorization for multi-relational learning. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 617–621, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[21] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428. International World Wide Web Conferences Steering Committee, 2016.

[22] M. E. Peters, M. Neumann, R. L. L. IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith. Knowledge enhanced contextual word representations, 2019.

[23] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[24] S. Ramprasad and J. Maddox. Coke: Word sense induction using contextualized knowledge embeddings. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2019.

[25] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[26] M. Salahli. An approach for measuring semantic relatedness between words via related terms. © *Association for Scientific Research*, 14:55–63, 04 2009.

[27] K. I. Simov, P. Osenova, and A. Popov. Comparison of word embeddings from different knowledge graphs. In *LDK*, 2017.

[28] N. A. Smith. Contextual word representations: A contextual introduction. *arXiv preprint arXiv:1902.06006*, 2019.

[29] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[30] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. Van Durme, S. R. Bowman, D. Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

[31] F. Torabi Asr, R. Zinkov, and M. Jones. Querying word embeddings for similarity and relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 675–684, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[32] F. Torabi Asr, R. Zinkov, and M. Jones. Querying word embeddings for similarity and relatedness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 675–684, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[33] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org, 2016.

[34] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledge base. 2014.

[35] B. Wang, A. Wang, F. Chen, Y. Wang, and C. J. Kuo. Evaluating word embedding models: Methods and experimental results. *CoRR*, abs/1901.09785, 2019.

[36] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, Dec 2017.

[37] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 1112–1119. AAAI Press, 2014.

[38] L. Wittgenstein, G. E. M. Anscombe, and L. Wittgenstein. *Philosophical Investigations... Translated by GEM Anscombe.(Philosophische Untersuchungen.) Eng. & Ger.* oxford, 1953.

[39] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[40] J. Xin, Y. Lin, Z. Liu, and M. Sun. Improving neural fine-grained entity typing with knowledge attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[41] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2014.

[42] M. Yu and M. Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 545–550, 2014.