WebBrain: Joint Neural Learning of Large-Scale Commonsense Knowledge

Jiaqiang Chen[1], Niket Tandon[2], Charles Darwis Hariman[3], and Gerard de Melo[4]

1: IIIS, Tsinghua University, Beijing, China**
2: Allen Institute for Artificial Intelligence, Seattle, WA, USA
3: Max Planck Institute for Informatics, Saarbrücken, Germany
4: Rutgers University, Piscataway, NJ, USA
http://gerard.demelo.org

Abstract. Despite the emergence and growth of numerous large knowledge graphs, many basic and important facts about our everyday world are not readily available on the Web. To address this, we present WebBrain, a new approach for harvesting commonsense knowledge that relies on joint learning from Web-scale data to fill gaps in the knowledge acquisition. We train a neural network model to learn relations based on large numbers of textual patterns found on the Web. At the same time, the model learns vector representations of general word semantics. This joint approach allows us to generalize beyond the explicitly extracted information. Experiments show that we can obtain representations of words that reflect their semantics, yet also allow us to capture conceptual relationships and commonsense knowledge.

1 Introduction

Motivation. In the past decade, massive amounts of machine-readable knowledge have become available, both in large knowledge graphs such as DBpedia, YAGO, and GeoNames, as well as through the widespread adoption of standards such as schema.org for Web pages. Additionally, information extraction techniques allow us to mine further knowledge from natural language text. To date, such data has mainly been used for improved information interchange and integration, e.g. for better Web search results on entity-focused queries or for novel kinds of visualizations that combine information from different sources.

However, while there are numerous bots and services that scour the Web to exploit a particular (often hard-coded) kind of information, we still lack intelligent systems that more flexibly draw advanced conclusions from information found on the Web. Among the missing ingredients, the lack of required world knowledge stands out as particularly relevant. This includes knowledge that is less of the factual, encyclopedic kind found in DBpedia, but more related to a general

** This work was supported in part by China 973 Program Grants 2011CBA00300, 2011CBA00301, NSFC Grants 61033001, 61361136003, 61550110504, and the Samsung R&D Institute of China.

understanding of everyday objects and concepts in the world. In some cases, such commonsense knowledge can be expressed as subject-predicate-object triples, similar to those used for factual knowledge. Relevant predicates include `causes` (e.g., fire causes heat), `hasProperty` (e.g., ice has the property of being cold and ice cream has the property of being sweet), `partOf` (e.g., legs as parts of humans), and `usedFor` (e.g., that keys can be used to open doors).

Goal. We ultimately aim at a system capable of guessing the truth of commonsense facts (e.g. whether a dog can fly), based on knowledge seen on the Web. However, procuring this sort of knowledge from the Web is non-trivial because shared assumptions about the world are often taken for granted such that it would be rare if not strikingly odd for someone to write that tractors are inedible or that radiologists are capable of breathing. Thus, information extraction alone is insufficient for equipping computational systems with commonsense knowledge.

Overview and Contribution. In this paper, we propose a joint learning approach to acquire commonsense knowledge both from explicit and implicit textual information. explicit triples and on large-scale word co-occurrence information. We optimize matrix representations of relations explicitly mined from large amounts Web data using a custom information extraction approach designed to minimize noise when applied to Web-scale data. At the same time, concepts are modeled as vectors trained on large-scale text following the word2vec CBOW approach to capture generic semantics [22].

As a result, our approach jointly learns representations of words and relations to better reflect our natural understanding of them. In particular, we are able to exploit general Web-scale semantics when learning commonsense relationships, e.g. inferring from eagles being capable of flying that hawks are likely also capable of the same. Our experiments show that we can obtain representations that simultaneously capture conceptual relationships as well as word meanings.

2 Background and Related Work

Commonsense knowledge acquisition has been studied for many years now. Traditionally, such knowledge was modeled by human experts, an approach best exemplified by the Cyc project [18], a decades-long commercial effort at creating a large axiomatic rule base. The SUMO ontology [24] shares this goal, but relies on open source principles and more collaborative development processes. However, in both cases, contributing requires significant expertise and effort in knowledge modeling. Although feasible for specific domains, it is difficult to obtain extensive amounts of commonsense knowledge in this way.

For large-scale commonsense knowledge acquisition, there are two more promising directions. The well-known ConceptNet project [12] relies on crowdsourcing, aiming at much simpler commonsense knowledge propositions. Another approach is to turn to large-scale data mining. Many information extraction papers follow the bootstrapping method proposed by Hearst [13], who used linguistic patterns to mine `isA` relationships. However, pattern-based approaches tend to extract only few facts and suffer from significant problems with noise. Several approaches

have been proposed to improve bootstrapping in general [25, 31]. Another route is to develop improved algorithms catering to particular kinds of infromation, e.g. temporal knowledge [9], properties and attributes [30], or activity knowledge [32].

Irrespective of whether one relies on crowdsourcing or data mining, however, it is necessary to generalize and expand beyond what has explicitly been acquired. For instance, we may have obtained that Samoyed dogs have fur but we may not have explicitly found this to be the case for shiba inus as well.

This leads us to the task of knowledge base completion [26]. When relying on machine learning, this typically becomes a relation prediction task. Given a a training set of true example instances of relations, i.e. triples, the goal is to learn a model that can then be used to predict whether a new, previously unseen triple is true or false. While the relation itself will have occurred in the training set, the specific triple will be new.

One approach is to consider this a tensor or matrix completion problem. For instance, if we view a relation as a matrix between subjects and objects storing their truth values, then relation prediction boils down to filling in the missing values to complete the matrix. Previous work in this area includes AnalogySpace [28], which relied on singular value decomposition applied to ConceptNet extractions. Nickel et al.'s RESCAL [23] uses tensor factorization to model relationships, targeting collective classification and entity resolution. Sutskever et al. [29] propose Bayesian clustered tensor factorization to model relational data. Jiang et al. [16] proposed a generative probabilistic model for relation prediction based not only on existing triples but also on information extraction.

A more recent line of work uses neural networks for relation prediction. Bordes et al. [4, 5, 15, 6] proposed several neural network architectures to capture relation triples, the most well-known of these being the TransE approach, which models the relation as a translation from a vector for the subject to a vector for the object. Numerous variations have been proposed that modify how the relation is modeled. For instance, Socher et al. [27] propose neural tensor networks (NTN), in which each relation is represented as a tensor. TransH [35] models relations as translations on hyperplanes. TransR [20] adds extra projections of entity vectors for each specific relation, or, in the CTransR variant, for each cluster of relations. PTransE [19] attempts to consider inference via property paths to improve the prediction of a triple (for example, x `bornInCity` y, y `cityInState` z helps us predict x `bornInState` z).

Our approach differs from all such previous efforts by learning to generalize not just based on the existing triples, as done by matrix and tensor methods as well as the TransE-related neural models, but by additionally exploiting semantic information derived from large-scale text statistics. As we show in our experiments, pure relational modeling does not result in semantically satisfactory word vector representations. Our joint model alleviates this problem by enabling the choice of word representations to benefit from large amounts of raw text, similar to the way humans draw on general semantic associations as well as more explicit information. Since the word vectors are constrained to reflect semantic

similarity, we use more flexible matrix representations of relations rather than simple translations as in the TransE model. Compared to the NTN model, in contrast, we model relations in a less flexible way, so as to ensure a mutual influence between commonsense relations and word representations. Compared to Jenatton et al. [15], which in turn is related to the NTN model, we do not use any 1-gram or 2-gram features, but directly use the product as a scoring function. We also forgo requiring that the matrix be the sum of rank-1 matrices. This enables our approach to scale to much larger data sets such as DBpedia.

Our joint model learns word representations that allow us to better transfer knowledge between related concepts. We rely on Mikolov et al.'s word2vec CBOW approach [22], who simplified previous neural language models [1] for significantly greater scalability. They also introduced the Skip-Gram model as an alternative, but in our approach, we build on the CBOW variant, as it is faster to optimize. There have been other proposals to adapt the word2vec models. Several approaches aim at improving word vectors using additional knowledge of semantic similarity [37, 8]. These are based on generic semantic similarity rather than capturing specific kinds of relations. Hill & Korhonen [14] presented a model for multi-modal representations, training on large amounts of image labels and text, with a minor addition of 638 abstract concept descriptions.

Bollegala et al. [3] proposed a method for obtaining improved word vectors by exploiting information about the lexical patterns they occur in. This approach is aimed at obtaining vectors that better reflect word analogies but does not address our model's goal of relation prediction. Xie et al. [36] exploit entity description glosses but do not use large-scale text. Wang et al. [34] proposed the *probabilistic TransE* model, capturing Freebase triples following the TransE model, but also viewing the co-occurrences of two phrases as a relationship that should likewise be modeled as a translation. Their model uses two vectors per phrase and an alignment component to connect entities to phrases. Our model uses just a single vector per word, so mutual dependencies between word vectors are exploited to a greater degree, while the relation modeling is less constrained due to the use of matrices, so a greater divergence from the word relationships is enabled. Toutanova et al. [33] also attempt to model relationships between two entities found in text, but use syntactic dependency trees and apply a convolutional neural network over them to obtain relation representations. In contrast, we exploit any occurrence of a word, not just explicit relationships between two entities in a sentence.

3 Web-Scale Knowledge Bootstrapping

Pattern-Based Information Extraction. For knowledge acquisition, it is well-known that one can attempt to induce patterns based on matches of seed facts, and then use pattern matches to mine new knowledge [13]. Unfortunately, this bootstrapping approach suffers from significant noise when applied to large Web-scale data [31], which appears necessary in order to mine adequate amounts of training data. Specifically, we rely on Google's large Web N-Gram dataset

(see Section 5). This problem is exacerbated by the fact that we are aiming at commonsense knowledge, which is not typically expressed using explicit relational phrases. We thus devise a custom bootstrapping approach designed to minimize noise when applied to Web-scale data.

We assume that we have a set of relations $\mathcal{R}$, a set of seed facts $S(r)$ for a given $r \in \mathcal{R}$, as well as a domain(r) and range(r), specified manually to provide the domain and range of a given r as its type signature. For pattern induction, we look for co-occurrences of words in the seed facts within the n-grams data (for $n = 3,4,5$). Any match is converted into a pattern based on the words between the two occurrences, e.g. *that apple is red* would become $\langle x \rangle$ *is* $\langle y \rangle$.

Pattern Scoring. The acquired patterns are still rather noisy. To score the reliability of patterns, we rely on a ranking function that rewards patterns with high distinct seed support but also discounts patterns that occur across multiple dissimilar relations [31]. The intuition is that a good pattern should match many of the seed facts, but should not be overly generic so as to apply to many relations (as, e.g., $\langle x \rangle$ *or* $\langle y \rangle$). A pattern with many matches for both `hasLocation` and `partOf` is less likely to be a reliable one.

Still, a pattern that matches `isa` or `hasLocation` may also match a relation such as `conceptuallyRelatedTo`. To allow for this, we first define a relatedness score between relations. We can either provide these scores manually, or consider Jaccard overlap statistics computed directly from the seed assertion data. Let p be a candidate pattern and $r \in \mathcal{R}$ be the relation under consideration. We define $|S(r,p)|$ as the number of distinct seeds $s \in S(r)$ under the relation r that p matches. We then define the score of the pattern p for relation r as:

$$\phi(r,p) = \sum_{r' \in \mathcal{R}, r' \neq r} \frac{|S(r,p)|}{|S(r)|} - (1 - \text{sim}(r,r')) \frac{|S(r',p)|}{|S(r')|}$$

where $\text{sim}(r,r')$ is the similarity between relations r and r'. At the end, we choose the top-k ranked patterns as the relevant patterns for the extraction phase.

Assertion Extraction. We apply the chosen patterns to find new occurrences in our (Google Web N-grams) data. For instance, $\langle x \rangle$ *is* $\langle y \rangle$ could match *the sun is bright*, yielding (*sun*, *bright*) as an assertion for the `hasProperty` relation. To filter out noise from these candidate assertions, we check if the extracted words match the required domain and range specification for the relation, using WordNet's hypernym taxonomy. Finally, we rank the candidate assertions analogously to the candidate patterns, but treating the patterns as seeds.

4 Representation Learning and Prediction

Figure 1 provides an overview of our approach. Having extracted triples from text, the next step is to train a model for learning commonsense word and relation representations. Our model takes the extracted knowledge and learns to generalize it by simultaneously drawing not only on the extractions but also on large amounts of generic text. For instance, we may have observed that pigeons

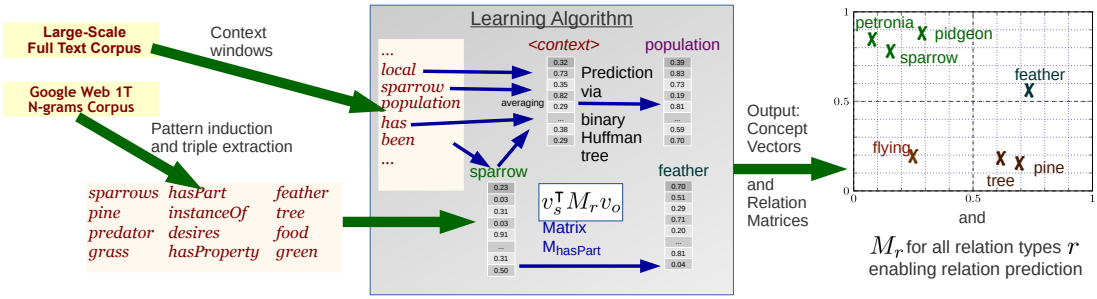

Fig. 1. Schematic overview of our approach.

fly but not that sparrows do. Our model aims to infer the latter based on both the extracted facts as well as general semantic relatedness between pigeons and sparrows. For the latter, we use word co-occurrences observed in large amounts of text. While word co-occurrences provide only weak signals of semantic relatedness, we can benefit from their large quantities. Thus, their overall contribution may make up for some of the sparsity of the explicitly extracted knowledge.

At the same time, word representations can also benefit from the joint learning setup. We learn word meanings not only from general context information, but also from the extracted relationships. For instance, we may have extracted that roses tend to have the property of being red, or that fire causes heat. From a cognitive perspective, commonsense knowledge about concepts is salient and should also guide the meaning representation of words. Distributional semantics has a long history, which has often been linked to J.R. Firth's famous quote that one shall "know a word by the company that it keeps". Still, while it is indisputable that regular contexts play a vital role in meaning acquisition, words and concepts are often acquired by other means than just from general contexts. Depending on the situation, humans may pay special attention to certain cognitively salient features and relationships of an object (e.g., appearance and function). In our approach, we thus train concept representations jointly with relationship representations, exploiting both the general contextual information and the mined relationship data.

This also touches on the long-standing dispute about whether mental representations of concepts are best modeled using discrete symbolic methods or in connectionist models based on numerical information processing. Whilst this has sometimes been regarded as an irreconcilable dichotomy, models like the one we propose here learn neural representations of words but also capture explicit symbolic relationships between them.

4.1 Objective

For the general word co-occurrence information, we adopt the word2vec CBOW objective [22]. The idea is to find vector representations of words such that the

surrounding words enable the prediction of a given target word. One maximizes

$$\sum_{w} \log P(w \mid C(w)), \tag{1}$$

where w denotes a word token in a large corpus and $C(w)$ denotes the context words. In the CBOW model, w is represented by a dense, real-valued word vector v_w and the context $C(w)$ is represented by the average of the vectors for the surrounding words. The resulting context vector is used to predict v_w.

Simultaneously, we jointly optimize for modeling the explicit relationships mined earlier. We assume that we have extracted labeled relationships between words. These can be viewed as (s, r, o) tuples consisting of a left word s (the subject), a predicate (relation) r, and a right word o (the object). We wish to use matrices and vectors to capture the information that the extracted relational data provides. We still assume every word is mapped to a vector, but additionally map each relation type to a specific matrix. To learn these representations, we seek to maximize a scoring function over all relation triples. We define

$$f(s, r, o) = v_s^\intercal M_r v_o,$$

where v_s is the word vector for the subject s of the relation triple and v_o is the word vector for the object o, while M_r is a matrix for relation r. If M_r is the identity matrix, then this function will compute a simple dot product measuring the similarity of the two vectors. If M_r is some other form of diagonal matrix, f would compute a weighted vector similarity. Other forms of M_r can capture transformations of the two vectors. The word vectors, described by v_s and v_o here, are jointly modified by both the CBOW and the relation modeling components, while the relation matrices M_r are only modified by the latter. For this relation modeling, we rely on the following loss function to quantify the error:

$$L_{s,r,o,l} = -l \log(\sigma(f(s, r, o))) - (1 - l) \log(1 - \sigma(f(s, r, o))), \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and l is the label of the training triple s,r,o (1 for positive training examples and 0 for negative ones).

Finally, we train our model to learn representations both from the relations and using the word2vec CBOW objective, to exploit the contextual statistics from large raw text corpora, thus making our representations more meaningful, as we will show later on in the experiments. Our overall loss function is as follows (with a parameter β to control the relative contributions):

$$\begin{aligned} &\sum_{(s,r,o,l)} -l \log(\sigma(f(s, r, o))) - (1 - l) \log(1 - \sigma(f(s, r, o))) \\ &+\beta \sum_{w} -\log P(w \mid C(w)) \end{aligned} \tag{3}$$

4.2 Training

During training, we seek to minimize this loss by simultaneously optimizing both parts of the objective. For the CBOW part, we follow the well-known negative

sampling procedure [22]. For the relation tuples, the training procedure is as follows. Given a training tuple, we generate k random negative examples by replacing the subject or object with a random word. Every triple is mapped to the corresponding word vectors and relation matrices. For each triple, both negative and positive ones, we optimize the loss function mentioned above.

For this optimization, we rely on stochastic gradient descent, which involves repeatedly picking random training examples, evaluating the current model on them, and making small updates if the model gives an incorrect answer. The direction of an update step is given by the gradient of the objective function, while the learning rate is a small factor that determines how much we move the model parameters (in our case, the values in the word vectors and relation matrices) in this direction. The gradients with respect to the second sum in Eq. 3 are as for the standard word2vec CBOW model, while for the first sum they are as follows:

$$\frac{\partial L_{s,r,o,l}}{\partial v_s} = (1-l)\, M_r v_o \quad \frac{\partial L_{s,r,o,l}}{\partial v_o} = (1-l)\, M_r^\intercal v_s \quad \frac{\partial L_{s,r,o,l}}{\partial M_r} = (1-l)\, v_s v_o^\intercal$$

Although individual updates with respect to the two parts of the objective function may pull the model in different directions, stochastic gradient descent normally finds stable solutions in the long run. In our case, this is expected because objects with similar extracted properties are also likely to be similar from a word semantics perspective. Our experimental results confirmed this.

5 Experiments

5.1 Data and Extraction

General Corpus. For our experiments, we rely on two datasets. The first is a frequently used dump of the English Wikipedia[1] that serves as our general corpus for word representation learning. We normalize the text to lower case and remove special characters, obtaining 1,205,009,210 tokens after this preprocessing.

Extraction Corpus. In order to extract relations, we turn to a Web-scale resource based on much larger quantities of text, the Google Web 1T N-gram dataset[2]. Although this data is limited to short 5-grams, it is well-suited for the kind of general commonsense knowledge relationships between words that we are targeting.

Seed Data. In order to bootstrap the extraction process, we rely on seed facts taken from the ConceptNet dataset [21] to induce patterns for each commonsense relation in ConceptNet. Examples of these relations include `atLocation`, `causes`, `hasProperty`, `motivatedByGoal`, `partOf`, and `usedFor`. This data is rather noisy, mostly due to incorrect natural language analysis of crowdsourced statements. We further find that more than 90% are named entities, since ConceptNet also imports

[1] `http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2`
[2] `https://catalog.ldc.upenn.edu/LDC2006T13`

from other existing knowledge sources. We remove these and also generally filter out all concepts not in WordNet, a lexicon containing mostly general words. At this point, we obtain a size of nearly 192K assertion triples. We then applied a score threshold of 5 (i.e., we enforce that at least five crowdsourcing annotators agree) to filter out further noise. Additionally, we required that the subject and object of each triple match our domain(r) and range(r) for the involved relation r. This is checked using the WordNet taxonomy [10]. For instance, for the `hasProperty` relation, we accept (*apple*, *red*), because *apple* is classified as a physical noun in WordNet and *red* as an adjective. A manual annotation of two random samples of size 200 revealed that the raw ConceptNet facts had an accuracy of only 53%, while the filtered seed facts had an accuracy of 99%.

5.2 Extraction

We then follow the extraction approach described in Section 3. Applying the seeds to our Web-scale n-grams, we obtain large numbers of patterns. Table 1 shows the top patterns for a few example relations.

These patterns then give rise to vast quantities of commonsense relation triples, each consisting of word pairs as well as the relation between them. We extract triples for 24 different relation types.

After filtering for noise we are left with a total of 1,160,136 extractions, e.g. (*abbey*, *church*) for the `isA` relation, or (*telephone*, *notice*) for the `usedFor` relation.[3]

Table 1. Top-k Patterns for some relations

AtLocation	IsA	UsedAs	MadeOf	HasProperty
X across Y	X was only Y	X used to Y	X made of Y	X is very Y
X inside Y	X except Y	X is used to Y	X repair Y	X can be Y
X outside Y	X called Y	X designed to Y	X is made of Y	X is too Y
X near Y	Y is X	X was used to Y	X made from Y	X may be Y
X under Y	X means any Y	X to help Y	X cast Y	X is as Y

Before training, we filter out triples that contain words appearing less than 100 times in Wikipedia and obtain 1,158,141 triple instances. We split the data and use 118,826 triples each for validation and testing, and the remaining ones for training. We also obtain a human-created gold dataset as ground truth, by taking a human-verified subset of ConceptNet with over 20,000 triples.

Our goal is to use this knowledge in order to train a neural network so as to learn word vectors that reflect semantic and commonsense knowledge, and to be able to generalize this to new commonsense not directly observed in the data.

[3] See `http://gerard.demelo.org/webbrain/`.

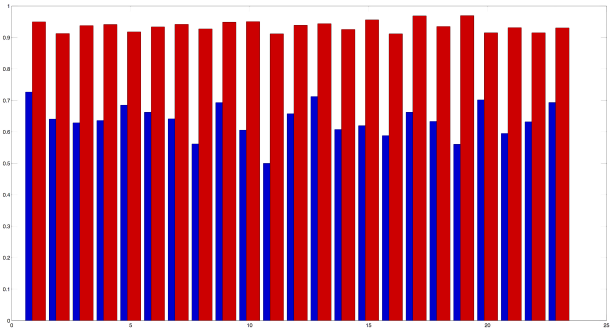

Fig. 2. Accuracy for every relation type comparing the fixed vector baseline (dark blue) and the joint training of WebBrain (brighter red).

This should enable the representations to capture cognitively salient information inherent or associated with word meanings, e.g. that a *tyre* is part of a *car*. We can train the word vector representations by jointly optimizing for the relations and optimizing for the word2vec CBOW model. Raw text like that from Wikipedia provides regular contexts, while the triples describe common sense relationships, thus contributing different kinds of information to the representations.

5.3 Training

We consider several experimental setups. In the first setup, we pre-initialize the word representations using vectors from the word2vec CBOW model, utilizing the information from the Wikipedia text corpus. To test if these representations alone can successfully be used to reflect the relations, in this first setup, we fix the embeddings during the training and just modify the relation matrices. The training proceeds for 10 iterations.

In the second setup, we pre-initialize the vectors in the same way but allow both the vectors and the relation matrices to be modified during the 10 training iterations. In the third setup, instead of pre-initializing the vectors, we train the relational data jointly with the word2vec CBOW model, optimizing both simultaneously. In all setups, we use a vector dimensionality of 50 in order to reduce the runtime. In the relation-only setups, following the literature, we normalize the word vectors during the training and use a standard learning rate of 0.01. For each training triple, we generate 5 negative examples by randomly replacing its left or right word with a random word in the vocabulary. The vocabulary is created with words appearing at least 100 times in the Wikipedia.

When we train the triples jointly with the word2vec objective, we optimize both objectives simultaneously until the CBOW architecture has completed 3 epochs. This is done in parallel in several threads that can run on multiple cores.

Alongside with several threads for the word2vec model, we create additional threads for our relational objective function from Section 4. This time, we do not normalize the word vectors in the relational thread, as we are training jointly and this would not make good use of the raw text. Instead of varying both β and the learning rates, we simply factor the variation of different possible choices of β into the choice of learning rate for the relational component. We fixed the CBOW learning rate at 0.05, while tuning the relational one on the validation data but also checking the WS-353 word similarity dataset. While there is no separate held-out dataset available for these word similarities, the much larger MEN dataset was not used for tuning. We describe these datasets in more detail later on. Ultimately, we arrived at the a much lower rate of just 0.002 for the relational data, which avoids distorting the vectors too much. With higher learning rates, we obtain almost the same results in terms of relation prediction, but the word vectors become overly biased towards those predictions and the word similarities correlate less strongly with human judgements.

For comparison, we also experiment with the TransE model as a representative example of methods that only use the existing relation triples without relying on information from large-scale text. Following the literature, we set the starting learning rate to 0.001 and require that the margin between positive triple and its corresponding negative samples be at least 1. We pre-initialize the model with the word2vec vectors and train it for 500 iterations, which suffices for convergence.

The final vectors and matrices successfully separate the positive training triples and randomly generated negative ones, as indicated in Figure 3. The y axis here reports the $v_s^\intercal M_r v_o$ scores. We can see that if we fix the word vectors and just optimize the matrix, the scores of positive and negative examples mix. If we allow the word vectors to change, the scores of the positive and negative examples are better separated.

5.4 Evaluation and Analysis

We attempt to discriminate between positive triples (from the test and gold sets) and random triples to assess whether our model can successfully capture the relations and classify unseen triples. In our model, the positive $v_s^\intercal M_r v_r$ scores are usually bigger than random ones, while in the TransE model, positive examples usually have smaller scores. We use the validation data set to choose the threshold. For our model, test examples with scores below the threshold are classified as negative and those with larger scores are classified as positive. For the TransE model, the opposite applies.

The classification results are presented in Table 2. Results on the test split reflect how well our model learns to predict the extractions, while results on the gold data set reflect to what extent the predicted relationships really hold true from a ground truth perspective. If we fix the vectors to be the ones from word2vec ("relations only, fixed vectors"), the accuracy is rather low, suggesting that the word2vec vectors are not well-suited for capturing relational similarities based on commonsense knowledge. However, when we allow our algorithm to modify the vectors ("relations only"), the resulting model achieves a good accuracy. We

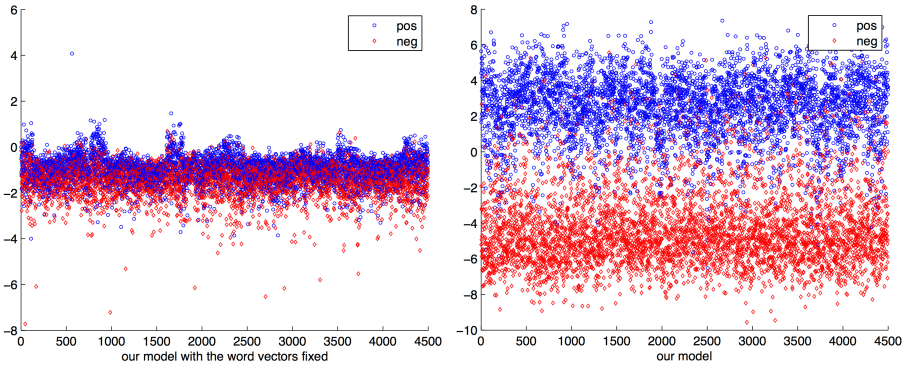

Fig. 3. Comparing positive examples in the validation set (dark blue circles) with negative ones (light red diamonds). Left: $v_s M_r v_o$ scores for WebBrain when vectors are fixed to the original vectors from word2vec. Right: scores for regular WebBrain.

Table 2. Relationship modeling results

Approach	Test	Gold
TransE	0.976	0.892
WebBrain: relations only, fixed vectors	0.761	0.700
WebBrain: relations only	0.983	0.897
WebBrain joint text+relation model	0.969	0.935

obtain 0.983 on the test data set and 0.897 for the gold, which is comparable with the TransE model despite our scalable training procedure. This setting corresponds to using only the first term of Eq. 3.

If we train our relational model jointly with the word2vec CBOW model, i.e. using the full objective given by Eq. 3, we see a slight reduction in accuracy on the test set, i.e. in predicting the original extractions. This is understandable, given that we are no longer optimizing for the goal of predicting relations exclusively. However, we obtain a significant improvement on the gold data set, showing that the model better reflects the real properties of concepts. This suggests that the joint training with general word semantics gives WebBrain better generalization capabilities than relation prediction models only considering the training triples.

In Figure 2, we see that the advantage of joint training is consistent across relation types. For each relation, we plot the fixed vector baseline (left, dark) and our joint training method (right, lighter). The 24 different relations are plotted along the x axis, while the y axis corresponds to the accuracy in $[0, 1]$.

Word Representations. We also evaluate the word representations directly, using two semantic relatedness datasets to assess the semantic similarities reflected

Table 3. Spearman's ρ for word similarity data

Approach	WS-353	MEN
WebBrain: text only	0.621	0.668
WebBrain: relations only	0.316	0.302
WebBrain joint text+relation model	0.632	0.679

in the word vectors. One is the well-known WS-353 [11] dataset, while the second is a significantly larger one called MEN [7]. Both contain English word pairs with similarity judgements elicited from human assessors. We calculate the cosine distance of word vectors for the word pairs in these datasets and compare them to the scores from the human annotations using Spearman's ρ.

Table 3 shows how the resulting word representations fare on the WS-353 and MEN datasets. For the vectors trained just on the relational data, e.g. with the TransE model, the result is significantly worse than for the text-only model. This means that, after training, the vectors are optimized for the relations and fail to reflect much of the information that the raw corpus data provides. However, if we train jointly, we observe better correlations, indicating that the vectors are able to maintain the semantic information from the raw corpus contexts.

This shows that our model can modify the word vectors to reflect commonsense relations without degrading the quality of general word similarities.

5.5 Additional Experiment on DBpedia

Data. We additionally evaluate the model's performance on DBpedia [2]. We focus on extractions from the English Wikipedia, using the "mapping-based types" and "mapping-based properties (cleaned)" data.

We consider only URIs from within DBpedia (starting with "http://dbpedia.org /resource/"), since others are not part of DBpedia itself and thus the data only contains very sparse, incomplete information about them. After preprocessing, we obtain a total of 15,109,444 triples describing 4,222,635 entities and 675 distinct relation types. We split this data by reserving 15,110 triples as validation data for tuning and 15,110 as a test data set for evaluation. All remaining triples are used as training data.

Training and Evaluation. To determine the optimal parameter settings, we rely on the validation data and choose the vector size for the entities from {30, 50, 100}. We run the experiment for {20, 40, 60, 80, 100} iterations. Based on these options, we select the best-performing set of parameters in terms of their accuracy on the validation data, as explained below. Following the literature, we normalize the vectors after every stochastic gradient descent step. For every iteration, we first shuffle the training triples. We decrease the learning rate linearly until it reaches 0.0001 of the starting learning rate. At that point, we hold it constant at that value, i.e. 0.0001 of the starting learning rate.

For comparison, we again also consider the TransE model on this data. We set the parameters for the TransE model as described in the original paper. The initial learning rate is set to 0.001, and the optimization proceeds for 1000 iterations. The vector size is also chosen from {30, 50, 100}.

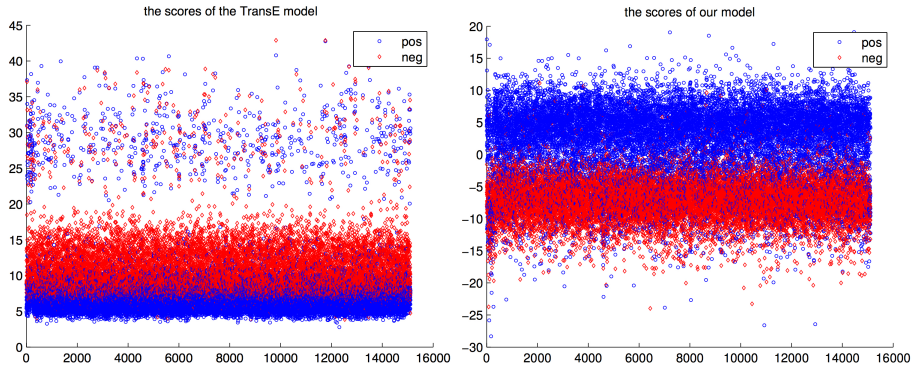

Fig. 4. Scores on the DBpedia validation data, computed as $\|v_s + v_r - v_o\|$ for the TransE model (left) and $v_s M_r v_o$ for ours (right), with legend as in Fig. 3.

We test the model's performance by discriminating between true and random triples. After the training, their classification scores should be different. True triples should have larger scores than random negative scores and we indeed observe this result. Fig. 4 plots the scores for the validation data set. We can see that the TransE model already achieves reasonably good results. Most positive triples have smaller scores. For our model, although we do not use a max-margin approach, the scores of positive and negative triples separate naturally as a result of the training objective. True triples usually have positive scores, while randomly generated ones have negative scores.

We use the validation data set to choose the best threshold that separates positive from negative triples and then test the model's ability to discriminate these on the test data set. For the TransE model, the best result is obtained with 50 dimensions and a threshold of 7.0, reaching an accuracy of 0.8245. For our model, setting the dimensionality to 50 and running for 60 iterations, the threshold is -2.26 and the accuracy is 0.8831. This shows that our model can successfully predict relationships even for the rich set of entities in DBpedia.

6 Conclusion

We have proposed WebBrain, a novel approach for knowledge acquisition and modeling, that makes a further step towards the goal of equipping computers with commonsense knowledge to enable more intelligent systems. Our model

combines multiple objectives to model word meanings and relationships. We rely on large-scale Web information extraction and on general corpus co-occurrences to train our model. While we remain far from genuine commonsense understanding and intelligence, our approach is able to learn vector representations of words and relations that reflect both their general semantics and basic commonsense facts about the world, giving accurate answers even for knowledge that has not been observed in text.

In future work, our joint training approach could also be evaluated with further relation representation models. This seems particularly promising for models that incorporate additional reasoning capabilities [19] or constraints [17]. Finally, we wish to extend our approach to combine common-sense knowledge as extracted from text with the kinds of encyclopedic facts available in DBpedia so as to obtain a more complete model of world knowledge. We believe that models of this sort that combine heterogeneous kinds of inputs will enable us to put semantic resources to use in advanced intelligent systems.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. 3, 1137–1155 (2003)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia – A crystallization point for the web of data. Web Semant. 7(3), 154–165 (2009)
3. Bollegala, D., Maehara, T., Kawarabayashi, K.i.: Embedding semantic relations into word representations. In: Proceedings of IJCAI. pp. 1222–1228 (2015)
4. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Proceedings of AAAI (2011)
5. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: Joint learning of words and meaning representations for open-text semantic parsing. In: Proceedings of AISTATS (2012)
6. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26, pp. 2787–2795 (2013)
7. Bruni, E., Tran, N.K., Baroni, M.: Multimodal distributional semantics. J. Artif. Int. Res. 49(1), 1–47 (2014)
8. Chen, J., Tandon, N., Gerard de Melo: Neural word representations from large-scale commonsense knowledge. In: Proceedings of WI/IAT 2015 (2015)
9. Espinosa, J., Lieberman, H.: EventNet: Inferring temporal relations between commonsense events. In: Proceedings of MICAI 2005, LNCS, vol. 3789. Springer (2005)
10. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. The MIT Press (1998)
11. Finkelstein, L., Evgenly, G., Yossi, M., Ehud, R., Zach, S., Gadi, W., Eytan, R.: Placing search in context: the concept revisited. In: Proceedings of WWW (2001)
12. Havasi, C., Speer, R., Alonso, J.: ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In: Proceedings of RANLP (2007)
13. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of COLING. pp. 539–545 (1992)
14. Hill, F., Korhonen, A.: Learning abstract concept embeddings from multi-modal data: Since you probably can't see what I mean. In: Proceedings of EMNLP. pp. 255–265 (2014)

15. Jenatton, R., Roux, N.L., Bordes, A., Obozinski, G.R.: A latent factor model for highly multi-relational data. In: Advances in Neural Information Processing Systems 25, pp. 3167–3175 (2012)
16. Jiang, X., Huang, Y., Nickel, M., Tresp, V.: Combining information extraction, deductive reasoning and machine learning for relation prediction. In: Proceedings of ESWC. Springer (2012)
17. Krompaß, D., Baier, S., Tresp, V.: Type-constrained representation learning in knowledge graphs. In: Proceedings of ISWC (2015)
18. Lenat, D.B.: CYC: A large-scale investment in knowledge infrastructure. Commun. ACM (1995)
19. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: Proceedings of EMNLP (2015)
20. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings AAAI 2015. AAAI Press (2015)
21. Liu, H., Singh, P.: ConceptNet—a practical commonsense reasoning tool-kit. BT Technology Journal 22(4), 211–226 (2004)
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26 (2013)
23. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of ICML (2011)
24. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proc. of FOIS (2001)
25. Pantel, P., Pennacchiotti, M.: Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In: Proceedings of COLING/ACL 2006 (2006)
26. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web (2016)
27. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems 26, pp. 926–934 (2013)
28. Speer, R., Havasi, C., Lieberman, H.: AnalogySpace: Reducing the dimensionality of common sense knowledge. In: Proceedings of AAAI. AAAI Press (2008)
29. Sutskever, I., Tenenbaum, J.B., Salakhutdinov, R.R.: Modelling relational data using Bayesian clustered tensor factorization. In: Advances in Neural Information Processing Systems 22, pp. 1821–1828 (2009)
30. Tandon, N., de Melo, G., Suchanek, F.M., Weikum, G.: WebChild: Harvesting and organizing commonsense knowledge from the web. In: Proceedings of WSDM (2014)
31. Tandon, N., de Melo, G., Weikum, G.: Deriving a Web-scale common sense fact database. In: Proceedings of AAAI 2011. AAAI Press, Palo Alto, CA, USA (2011)
32. Tandon, N., Weikum, G., Melo, G.d., De, A.: Lights, camera, action: Knowledge extraction from movie scripts. In: Proceedings of WWW (2015)
33. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of EMNLP. ACL (2015)
34. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph and text jointly embedding. In: Proceedings of EMNLP. pp. 1591–1601 (2014)
35. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of AAAI 2014. pp. 1112–1119 (2014)
36. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Proceedings of AAAI (2016)
37. Yu, M., Dredze, M.: Improving lexical embeddings with semantic knowledge. In: Proceedings of ACL 2014 (2014)