SeVeN: Augmenting Word Embeddings with Unsupervised Relation Vectors

Luis Espinosa-Anke and Steven Schockaert
School of Computer Science and Informatics, Cardiff University, UK
{Espinosa-AnkeL,SchockaertS1}@cardiff.ac.uk

Abstract

We present SeVeN (Semantic Vector Networks), a hybrid resource that encodes relationships between words in the form of a graph. Different from traditional semantic networks, these relations are represented as vectors in a continuous vector space. We propose a simple pipeline for learning such relation vectors, which is based on word vector averaging in combination with an *ad hoc* autoencoder. We show that by explicitly encoding relational information in a dedicated vector space we can capture aspects of word meaning that are complementary to what is captured by word embeddings. For example, by examining clusters of relation vectors, we observe that relational similarities can be identified at a more abstract level than with traditional word vector differences. Finally, we test the effectiveness of semantic vector networks in two tasks: measuring word similarity and neural text categorization. SeVeN is available at bitbucket.org/luisespinosa/seven.

1 Introduction

Word embedding models such as Skip-gram (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) use fixed-dimensional vectors to represent the meaning of words. These word vectors essentially capture a kind of similarity structure, which has proven to be useful in a wide range of Natural Language Processing (NLP) tasks. Today, one of the major applications of word embeddings is their interaction with neural network architectures, enabling a kind of generalization beyond those words that were only observed during training. For example, if a classification model has learned that news stories containing words such as 'cinema', 'restaurant' and 'zoo' tend to be categorized as 'entertainment', it may predict this latter label also for stories about theme parks due to the shared semantic properties encoded in word vectors. Word embeddings thus endow neural models with some form of world knowledge, without which they would be far less effective. This has prompted a prolific line of research focused on improving word embeddings not only with algorithmic sophistication, but also via explicit incorporation of external knowledge sources such as WordNet (Faruqui et al., 2015), BabelNet (Camacho-Collados et al., 2016) or ConceptNet (Speer et al., 2016).

Regardless of how word vectors are learned, however, the use of fixed-dimensional representations constrains the kind of knowledge they can encode. Essentially, we can think of a word vector as a compact encoding of the salient attributes of the given word. For instance, the vector representation of *lion* might implicitly encode that this word is a noun, and that lions have attributes such as 'dangerous', 'predator' and 'carnivorous'. Beyond these properties, word embeddings can also encode *relational knowledge*. For instance, the embedding might tell us that the words 'lion' and 'zebra' are semantically related, which together with the attributional knowledge that lions are predators and zebras are prey may allow us to plausibly infer that 'lions eat zebras'. However, the way in which relational knowledge can be encoded in word embeddings is inherently limited. One issue is that only relationships which are sufficiently salient can affect the vector representations of their arguments; e.g. the fact that Trump has visited France is perhaps not important enough to be encoded in the embeddings of the words 'Trump' and 'France' (i.e. there may be insufficient corpus-based evidence on this fact). Note that this is not a matter of how the embedding is learned; forcing the vector representations to encode this fact would distort the similarity structure of the embedding. From a formal point of view, there are also severe limitations to what can be encoded (Gutiérrez-Basulto and Schockaert, 2018). As a simple example, methods based on vector translations cannot model symmetric relations, and they are limited in the kind of many-to-many relations that can be encoded (Lin et al., 2015).

Like word embeddings, semantic networks such as WordNet (Miller, 1995), BabelNet (Navigli and Ponzetto, 2012) or ConceptNet (Speer et al., 2016) also encode lexical and world knowledge. They use a graph representation in which nodes correspond to words, phrases, entities or word senses. Edge labels are typically chosen from a small set of discrete and well-defined lexical and ontological relationships. Compared to word embeddings, the knowledge captured in such resources is more explicit, and more focused on relational knowledge (although attributional knowledge can be encoded as well, e.g., by using edge labels modeling the `has_property` relation). The use of discrete labels for encoding relation types, however, makes such representations too coarse-grained for many applications (e.g., a large proportion of the edges in ConceptNet are labelled with the generic 'related to' relationship). It also means that subjective knowledge cannot be modeled in an adequate way (e.g., forcing us to make a hard choice between which animals are considered to have the property 'dangerous' and which ones do not).

In this paper, we propose a hybrid representation, which we call SeVeN (Semantic Vector Networks). Similar to semantic networks, we use a graph based representation in which nodes are associated with words. In contrast to semantic networks, however, these edges are labelled with a vector, meaning that relation types are modeled in a continuous space.

To obtain a suitable *relation vector* for two given words a and b, we start by averaging the vector representations (from a pre-trained word embedding) of the words that appear in sentences that mention both a and b. The resulting vectors have two main disadvantages, however. First, they are high-dimensional, as they are constructed as the concatenation of several averaged word vectors. Second, the relation vectors are influenced by words that describe the relationship between a and b, but also by words that rather relate to the individual words a or b (as well as some non-informative words). Intuitively we want to obtain a vector representation which only reflects the words that relate to the relationship. For example, the relation vector for (paris,france) should ideally be the same as the vector for (rome,italy), but this will not be the case for the averaged word vectors, as the former relation vector will also reflect the fact that these words represent places and that they relate to France. To address both issues, we introduce an autoencoder architecture in which the input to the decoder comprises both the encoded relation vector and the word vectors for a and b. By explicitly feeding the word vectors for a and b into the decoder, we effectively encourage the encoder to focus on words that describe the relationship between a and b.

Once the semantic vector network has been learned, it can be used in various ways. For instance, the relation vectors could be used for measuring relational similarity (Jurgens et al., 2012), for identifying words that have a specific lexical relationship such as hypernyms (Vylomova et al., 2016), or complementing open information extraction systems (Delli Bovi et al., 2015). In this paper, however, we will assess the potential of SeVeN in terms of two tasks, namely using it for (1) unsupervised semantic similarity modeling, and for (2) enriching word vectors as the input to neural network architectures. The overarching idea in the latter case is that, instead of simply representing each word by its vector representation, the representation for each word position will be composed of (i) the vector representation of the word, (ii) the vector representations of the adjacent words in the semantic vector network, and (iii) the corresponding relation vectors (i.e. the edge labels).

2 Related Work

Related work broadly falls in two categories: methods which aim to improve word embeddings using relational knowledge, and methods which aim to learn relation vectors. To the best of our knowledge, there is no previous work which uses relation vectors with the aim of enriching word embeddings.

Improving Word Embeddings. One of the most notable features of word embedding models, such as

Skip-gram (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014), is the fact that various syntactic and semantic relationships approximately correspond to vector translations. One limitation of vector translations is that they are not well-suited for modeling transitive relations, which is problematic among others for the is-a relationship. To this end, a number of alternative vector space representations have been proposed, which are specifically aimed at modeling taxonomic relationships (Vendrov et al., 2016; Yu et al., 2015; Nickel and Kiela, 2017). Note that while such alternative embedding spaces can solve some of the limitations of standard embeddings w.r.t. modeling taxonomic relationships, there are many other types of relations that cannot be faithfully modeled in these representations. Moreover, these alternative embeddings are not necessarily well-suited for modeling word similarity. More generally, various authors have explored the idea of adapting word embeddings to fit the needs of specific tasks, e.g. aiming to make embeddings better suited for capturing antonyms (Ono et al., 2015), hypernyms (Vulic and Mrksic, 2017) or sentiment (Tang et al., 2014).

As mentioned in the introduction, the use of semantic networks for improving word embeddings, based on the idea that words which are similar in the semantic network should have a similar embedding, has been explored by various authors (Faruqui et al., 2015; Camacho-Collados et al., 2016; Speer et al., 2016). Another possibility is to use a semantic network to decompose word embeddings into sense embeddings by imposing the constraint that the word vector is a convex combination of the corresponding sense vectors, as well as forcing similarity of the sense vector with the vector representations of its neighbors in the semantic network (Johansson and Piña, 2015). Finally, let us refer to work that learns additional embeddings that coexist in the same space as lexemes, e.g., WordNet synsets (Rothe and Schütze, 2015) or BabelNet synsets (Mancini et al., 2017).

Relation Vectors The idea of learning a relation vector for two words a and b, based on the words that appear in their context, goes back at least to the Latent Relational Analysis (LRA) method from (Turney, 2005). In that work, a matrix is constructed with one row for each considered word pair, where columns correspond to lexical patterns that have been extracted from sentences containing these words. The relation vectors are then obtained by applying Singular Value Decomposition (SVD) on that matrix. Along similar lines, in (Riedel et al., 2013) relation vectors are learned by factorizing a matrix whose rows correspond to entity pairs and whose columns correspond to properties (in this case comprising both lexical patterns from a corpus and triples from a knowledge graph). More recently, several methods have been proposed that learn a vector describing the relationship between two words by averaging the embeddings of the words that appear in between them in a given corpus (Weston et al., 2013; Hashimoto et al., 2015; Fan et al., 2015), or by learning a vector representation from PMI-like statistics on how strongly different words are associated with the considered word pair (Jameel et al., 2017). Beyond these unsupervised methods, a wide variety of supervised neural network based architectures have been proposed for learning relation vectors that are predictive of a given relation type (Zeng et al., 2014; dos Santos et al., 2015; Xu et al., 2015).

3 Constructing Semantic Vector Networks

Our aim is to construct a graph whose nodes correspond to words, whose edges indicate which words are related, and whose edge labels are vectors that encode the specific relationship between the corresponding words. We will refer to this representation as a semantic vector network. In this section, we describe our methodology for constructing such semantic vector networks. First, in Section 3.1, we provide details about the source corpus and explain how the structure of the network is chosen. In Section 3.2 we then discuss how suitable relation vectors can be constructed.

3.1 Defining the Network Structure

Our source corpus is a dump of the English Wikipedia from January 2018. We opted to keep preprocessing at a minimum to ensure that any emergent linguistic or relational regularity is captured during the network construction stages. Specifically, we applied sentence segmentation and word tokenization using *nltk*[1]. We also single-tokenized multiword expressions based on several lexicons (Schneider et al.,

[1]nltk.org

2014), and finally removed stopwords using the CoreNLP list[2]. After the above steps, we selected the 10^5 most frequent words as our vocabulary. To determine which words should be connected with an edge, we rely on Pointwise Mutual Information (PMI), which measures the strength of association between two random variables. It is commonly used in NLP as a method for identifying related words, e.g. in factorization based methods for learning word embeddings (Turney and Pantel, 2010). Specifically, we express the strength of association between words w_i and w_j as follows:

$$\mathit{pmi}(w_i, w_j) = \log\left(\frac{x_{ij}x_*}{x_i x_j}\right)$$

In our case, x_{ij} is the number of times word w_i appears near word w_j, weighted by the nearness of their co-occurrences, and $x_i = \sum_j$, $x_{ij} = \sum_j x_{ji}$ and $x_* = \sum_i \sum_j x_{ij}$. Specifically, let I_{w_i} be the word positions in the corpus at which w_i occurs, then we define:

$$x_{ij} = \sum_{p \in I_{w_i}} \sum_{q \in I_{w_j}} n(p, q)$$

where $n(p, q) = 0$ if the word positions p and q belong to a different sentence, or if $|p - q| > 10$, i.e. if there are at least 10 words in between them. Otherwise we define $n(p, q) = \frac{1}{|p-q|}$.

sorrow		**tournament**		**videogame**		**riverbank**	
ppmi	w2v	ppmi	w2v	ppmi	w2v	ppmi	w2v
contrition	sadness	scotties	tourney	lego	videogames	danube	riverbanks
lamentation	anguish	double-elimination	tournaments	consoles	videogaming	erosion	river
woe	grief	single-elimination	Tournament	villains	next_gen_consoles	laboratories	creek
savior	profound_sorrow	pre-olympic	tournment	arcade	Videogame	opposite	riverbed
everlasting	deepest_sorrow	4-day	tourament	sega	gamers	vegetation	riverside
anguish	heartfelt_sorrow	eight-team	tourneys	ea	MMOG	tales	lake
grief	profound_sadness	winnings	touranment	playstation	PS2	washed	shoreline

Table 1: Examples of the highest scoring (i.e. most strongly associated by pmi) words, as well as their nearest neighbors in the pretrained word2vec (w2v) Google news vector space.

To choose the edges of the semantic vector network, we only consider word pairs which co-occur at least 10 times in the corpus. Among such pairs, for each word w_i, we first select the 10 words w_j whose score $\mathit{pmi}(w_i, w_j)$ is highest. This resulted in a total of about 900 000 pairs. Then, we added the overall highest scoring pairs (w_i, w_j) which had not yet been selected, until we ended up with a total of approximately 10^6 edges involving the initial vocabulary of 10^5 words. In the following we will write N_w for the neighbors of w, i.e. the set of words n such that $\{w, n\}$ was selected as an edge.

Note that by capturing pairs of words strongly connected by PMI, we encode a different type of relatedness than proximity in word embeddings. To illustrate this, in Table 1 we compare the most closely related words in our PMI graph, for some selected target words, with their nearest neighbors in the *word2vec* Google News word embedding space[3] (measured by cosine similarity). While the word2vec neighbors mostly consist of near-synonyms and other syntagmatic relationships, the chosen PMI pairs include a wide variety of topically related linguistic items. A semantic network based on such PMI pairs should thus capture information which is complementary to what is captured in word embeddings.

3.2 Learning relation vectors

Our general strategy for learning relation vectors is based on averaging word vectors. Specifically, for each sentence s in which w_i occurs before w_j (within a distance of at most 10), we construct three vectors, based on the words $a_1, ..., a_k$ which appear before w_i, the words $b_1, ..., b_l$ which appear in between w_i

[2]github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt

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

and w_j and the words $c_1, ..., c_q$ which appear after w_j:

$$pre^s_{w_i w_j} = \frac{1}{k}\sum_{r=1}^{k} \mathbf{v}_{a_r} \qquad mid^s_{w_i w_j} = \frac{1}{l}\sum_{r=1}^{l} \mathbf{v}_{b_r} \qquad post^s_{w_i w_j} = \frac{1}{q}\sum_{r=1}^{q} \mathbf{v}_{c_r}$$

where we write $\mathbf{v}_w$ for the vector representation of the word w. These vectors are then averaged over all sentences S_{ij} where w_i occurs before w_j:

$$pre_{w_i w_j} = \frac{1}{|S_{ij}|}\sum_{s\in S_{ij}} pre^s_{w_i w_j} \quad mid_{w_i w_j} = \frac{1}{|S_{ij}|}\sum_{s\in S_{ij}} mid^s_{w_i w_j} \quad post_{w_i w_j} = \frac{1}{|S_{ij}|}\sum_{s\in S_{ij}} post^s_{w_i w_j}$$

Since we can similarly obtain such vectors from sentences where w_j appears before w_i, we end up with a relation vector whose dimensionality is six times higher than the dimensionality of the word embedding, which would be impractical in the kind of applications we envisage (see Section 4). Another problem with these vectors is that they do not only reflect the relationship between w_i and w_j, but also the words w_i and w_j themselves. For instance, suppose we want to model the relationship between the words 'movie' and 'popcorn'. A sentence mentioning these two words could be:

Buttered popcorn is commonly eaten at movie theatres.

The most relevant words for describing the relationship are 'eaten at'. In contrast, however, 'buttered' is mostly related to the word 'popcorn' itself rather than describing its relationship with 'movie'. Similarly, 'theatres' is related to 'movie', but not relevant for characterizing the relationship.

To solve both issues, we propose to use an autoencoder architecture, in which the decoder has access to the word vectors $\mathbf{v}_{w_i}$ and $\mathbf{v}_{w_j}$, in addition to the encoded version of the relation vector. Let us write $\mathbf{z}_{w_i w_j}$ for the concatenation of $pre_{w_i w_j}$, $mid_{w_i w_j}$, $post_{w_i w_j}$, $pre_{w_j w_i}$, $mid_{w_j w_i}$, $post_{w_j w_i}$. Then the encoder is given by:

$$\mathbf{r}_{w_i w_j} = A\,\mathbf{z}_{w_i w_j} + \mathbf{b}$$

where $A \in \mathbb{R}^m \times \mathbb{R}^{6d}$ and $\mathbf{b} \in \mathbb{R}^m$, where d is the dimensionality of the word vectors, and m is the dimensionality of the encoded relation vectors m. In our experiments we experiment with different values for m. Empirically, we find that as the dimensionality of the compressed representations becomes smaller, the importance of word semantics gradually fades away in favor of their corresponding relational properties. Then, the decoder is then defined as:

$$\mathbf{z}^*_{w_i w_j} = B(\mathbf{v}_{w_i} \oplus \mathbf{r}_{w_i w_j} \oplus \mathbf{v}_{w_j}) + \mathbf{c}$$

where $\oplus$ denotes vector concatenation, $B \in \mathbb{R}^{6d} \times \mathbb{R}^{m+2d}$ and $\mathbf{c} \in \mathbb{R}^{6d}$. To train the autoencoder, we use the following L2-regularized reconstruction loss:

$$\mathcal{L} = \|\mathbf{z}_{w_i w_j} - \mathbf{z}^*_{w_i w_j}\|_2^2 + \lambda \|\mathbf{r}_{w_1 n_1}\|_2^2$$

with $\lambda > 0$ a regularization parameter. This loss function balances two objectives: minimizing the reconstruction error and keeping the L2 norms of the encoded relation vectors as small as possible. Because of this latter part, we can think of the norm of the relation vectors $\mathbf{r}_{w_i w_j}$ as a measure of how strongly the words w_i and w_j are related. In particular, if sentences mentioning w_i and w_j contain few or no words that describe their relationship, we might expect $\mathbf{r}_{w_i w_j}$ to be close to the 0 vector.[4]

4 Evaluation

We propose to evaluate our semantic vector networks from three different standpoints. First, we provide a qualitative evaluation by exploring relation network spaces (both compressed and uncompressed) and discussing meaningful properties. Second, we perform experiments in word similarity where we compare

[4]We also conducted experiments without the regularization term, with slightly worse results across all evaluations.

against the standard approach of measuring the similarity of two words by means of the cosine distance between their corresponding word vectors. This evaluation serves as an illustration of how semantic vector networks could be used in an unsupervised application setting. Third, as a prototypical example of a supervised application setting, we analyze the impact of leveraging the enriched representation these networks provide in neural text classification, in particular topic categorization and sentiment analysis. In all experiments, the pretrained embeddings we use (both for baselines and for constructing the relation networks) are the *word2vec* Google News embeddings (Mikolov et al., 2013b).

4.1 Qualitative Evaluation

One of the strongest selling points of word embeddings is that they enable inference of relational properties, which can be obtained by simple vector arithmetic such as summation and subtraction (Levy et al., 2015). The basic idea is that the relationship between two words w_i and w_j is characterized by the vector difference $\mathbf{v}_{w_i} - \mathbf{v}_{w_j}$. Such vector differences, however, encode relations in a noisy way. For instance, while the differences $\mathbf{v}_{rome} - \mathbf{v}_{italy}$, $\mathbf{v}_{paris} - \mathbf{v}_{france}$ and $\mathbf{v}_{dublin} - \mathbf{v}_{ireland}$ are all rather similar, there are in fact many other word pairs (not in a capital-of relationship) whose difference is also similar to these differences (Bouraoui et al., 2018). Accordingly, it was found in (Vylomova et al., 2016) that a relation classifier which is trained on word vector differences is prone to predicting many false positives. In contrast, we can expect that our relation vectors are modeling relations in a far less ambiguous way. On the other hand, these relation vectors are limited to word pairs that co-occur sufficiently frequently. Apart from the associated sparsity issues, this also suggests that relation vectors are not suitable for characterizing syntagmatic relationships and several types of syntactic relationships. We thus view these relation vectors as complementary to word vector differences.

In this section we illustrate the semantic properties of different versions of SeVeN. To this end, we show the nearest neighbors of selected target relation vectors for a number of different representations: (1) the original 1800d SeVeN network (`original`), (2) an autoencoded 10-dimensional space (`compressed-10d`), (3) a slightly higher-dimensional version (`compressed-50d`), and finally (4) a baseline model according to which the relation between two words is modeled as the vector difference of the corresponding word vectors (`diffvec`). The five selected target relation vectors, along with their nearest neighbors, are shown in Table 2. These target relation vectors were chosen to capture a range of different types of relationships, including hypernymic (‘nintendo - console’ and ‘gmail - email’) and attributional (‘roman - numerals’) relations.

One immediate observation is that, in most cases, the `diffvec` neighbors remain very close to the given word pair, where each word from the given pair is either preserved or replaced by a closely related word. The `original` and `compressed-50d` relation vectors largely follow a similar trend, although a few more interesting analogies are also found in these cases (e.g. *arabic - alphabet* as a neighbor of *roman - numerals*). The results for the `compressed-10d` vectors, however, follow a markedly different pattern. For these low-dimensional vectors, our autoencoder forces the relation vectors to focus on modeling the relationship between the two words, while abstracting away from the initial domain. This leads to several interesting neighbors, although this seems to come at the cost of some added noise.

Let us now analyze more closely the results of the `compressed-10d` vectors. If we read the first example along the lines of “juice can be made from limes”, similar relations are found close in the space, such as ‘coconut - milk’ and ‘marzipan - paste’. Note that the relation ‘noodles - egg’ is also similar, although the two words appear in the incorrect order (i.e. noodles can be made from eggs rather than the other way around). As another example where the directionality of this pattern is not captured correctly, we also find the pair ‘juice - lime’. It would be interesting to analyze in future work whether such issues can be avoided by using features from a dependency parser, e.g. following a similar strategy as in (Levy and Goldberg, 2014). Note that while all the `compressed-10d` neighbors are still related to food, these vectors have generalized beyond the domain of citrus fruits (see e.g., ‘lime’, ‘tamarind’ or ‘lemon’ in `diffvec`, or ‘lemon’ and ‘orange’ in `original`). A similar phenomenon occurs in some of the other examples. In the ‘nintendo-console’ case, after interpreting the relation as “major supplier of” or “entity which popularized”, we find nearest neighbors in the `compressed-10d` space

lime_juice

`original`	`compressed-10d`	`compressed-50d`	`diffvec`
lemon_juice	lemon_juice	lemon_juice	lime_soda
juice_lemon	coconut_milk	juice_lemon	lime_lemon
juice_lime	marzipan_paste	juice_lime	lemon_juice
lime_lemon	juice_lime	vinegar_sour	citric_juice
lemon_lime	noodles_egg	lemon_lime	tamarind_juice
pineapple_juice	lime_lemon	vinegar_sauce	lime_pie
orange_juice	marinated_beef	lime_lemon	pineapple_juice

nintendo_console

`original`	`compressed-10d`	`compressed-50d`	`diffvec`
wii_console	wii_console	wii_console	nintendo_consoles
playstation_console	playstation_console	nintendo_nes	nintendo_handheld
nintendo_nes	nintendo_nes	playstation_console	gamecube_console
xbox_console	witcher_2	xbox_console	wii_console
nintendo_consoles	itunes_download	nintendo_consoles	dreamcast_console
famicom_console	imax_2d	sega_consoles	nintendo_switch
nintendo_64	netflix_streaming	nintendo_handheld	3ds_console

gmail_email

`original`	`compressed-10d`	`compressed-50d`	`diffvec`
yahoo_email	renders_firefox	yahoo_email	gmail_emails
inbox_email	ie_browser	gmail_e-mail	yahoo_email
hotmail_email	infinitive_suffix	inbox_email	hotmail_email
email_yahoo	firefox_browser	gmail_emails	addy_email
gmail_e-mail	carnap_semantics	email_yahoo	imap_email
sending_email	helvetica_font	hotmail_email	smtp_email
send_email	cv_syllable	google_search	bugzilla_email

roman_numerals

`original`	`compressed-10d`	`compressed-50d`	`diffvec`
arabic_numerals	arabic_alphabet	arabic_numerals	cyrillic_numerals
letters_numerals	greek_alphabet	letters_numerals	indic_numerals
letters_alphabet	10-inch_discs	uppercase_letters	georgian_numerals
lowercase_letters	latin_alphabet	lowercase_letters	hieratic_numerals
arabic_alphabet	yemenite_pronunciation	uppercase_characters	brahmi_numerals
latin_alphabet	standard_orthography	latin_alphabet	sinhala_numerals
symbols_numerals	wii_remote	alphabetic_numerals	quantifiers_numerals

heavy_metal

`original`	`compressed-10d`	`compressed-50d`	`diffvec`
thrash_metal	metal_heavy	thrash_metal	heavy_metals
glam_metal	karma_dharma	doom_metal	cky_metal
doom_metal	techno_rave	glam_metal	manilla_metal
symphonic_metal	psychedelic_garage	thrash_slayer	annihilator_metal
nu_metal	cooking_recipes	punk_rock	heaviness_metal
sludge_metal	gita_yoga	hardcore_punk	doro_metal
glam_rock	post-punk_punk	sludge_metal	behemoth_metal

Table 2: Nearest neighbors (by cosine) for selected relation vectors and the three models under consideration.

where the same relation holds, but which do not belong to the video games domain, such as 'itunes-download' or 'netflix-streaming'. Next, we find that the relation holding between 'gmail' and 'email' is similar to 'ie' and 'firefox' and 'browser', and even 'helvetica' and 'font'. The relation between 'google' and 'search', found for `compressed-50d`, is also of this kind. In contrast, the `diffvec` neighbors in this case all have *email* as the second word. In the 'roman-numerals' example, likewise, the `diffvec` neighbors similarly have 'numerals' as the second word, while for `compressed-10d` we see more interesting neighbors such as 'arabic-alphabet' and 'yemenite-pronunciation'. We also find the seemingly unrelated 'wii-remote' pair, although we may consider that the Nintendo Wii console introduced a fundamentally new type of remote, which at an abstract level is similar to the fact that the Romans introduced a fundamentally new way of writing numbers. This example also suggests, however, that the way in which relations are modeled in the 10-dimensional space might be too abstract for some applications. Finally, the 'heavy-metal' case is a paramount example of how the the relation vectors may capture information which is fundamentally different than what is encoded by word vectors. In particular, the `diffvec` vectors all express relationships from the metalwork domain (e.g., 'heavy-metals' or 'annihilator-metal'), which reflects the fact that the music-related interpretation of the word 'metal' is not its dominant sense. In contrast, since our relation vectors are exclusively learned from sentences where both words co-occur ('heavy' and 'metal' in this example), the vector for 'heavy metal' clearly captures the musical sense (see e.g., 'thrash-metal' or 'glam-metal' in the `original` space).

4.2 Modeling Similarity

The capacity to capture and *embed* nuances of word meaning is one of the most celebrated features of word embeddings. The task of semantic similarity measurement, therefore, has been adopted as a *de-facto* testbed for measuring the quality of representations of linguistic items. The standard practice is to consider a distance (or similarity) metric such as cosine similarity and compare the similarity in a given vector space model with respect to human judgement. We note, however, that there exist other similarity metrics discussed in the literature, e.g., Weighted Overlap (Pilehvar et al., 2013) or Tanimoto Distance (Iacobacci et al., 2015). Our proposed similarity measurement parts ways from the idea of improving the representation of individual words, and rather seeks to refine their meaning by incorporating complementary cues via relation vectors, as well as the corresponding neighborhood structure. There are many possible ways in which this could be done, but we restrict ourselves here to a simple strategy, based on identifying the closest neighbors of the two words w_1 and w_2. The main intuition is that when w_1 and w_2 are similar, they should also be related to similar words. Specifically, we first determine the closest match between the neighbors of w_1 and the neighbors of w_2, as follows:

$$(n_1, n_2) = \underset{(n_1,n_2)\in N_{w_1}\times N_{w_2}}{\arg\max} \cos(\mathbf{v}_{n_1}, \mathbf{v}_{n_2}) + \cos(\mathbf{r}_{w_1 n_1}, \mathbf{r}_{w_2 n_2})$$

Note that to identify these neighbors, we compare both their word vectors $\mathbf{v}_{n_1}$ and $\mathbf{v}_{n_2}$, and their relationships to the target words, $\mathbf{r}_{w_1 n_1}$ and $\mathbf{r}_{w_2 n_2}$. Once these neighbors have been identified, we compute the similarity between w_1 and w_2 as follows:

$$\mathit{sim}(w_1, w_2) = \cos(\mathbf{v}_{w_1} \oplus \mu\mathbf{v}_{n_1} \oplus \mathbf{r}_{w_1 n_1}, \mathbf{v}_{w_2} \oplus \mu\mathbf{v}_{n_2} \oplus \mathbf{r}_{w_2 n_2})$$

where $0 < \mu \leq 1$ is a scaling factor which is aimed at reducing the impact of the neighbors n_1 and n_2 on the overall similarity computation. The fact that $\mathbf{v}_{n_1}$ is similar to $\mathbf{v}_{n_2}$ is an important indicator for the similarity between w_1 and w_2, but it should not influence the resulting similarity score as much as the similarity of the word vectors of w_1 and w_2 themselves. Rather than tuning this value, in the experiments we have fixed it as $\mu = 0.5$, which was found to give better results than $\mu = 1$ (i.e. no scaling). Note that the proposed way of computing similarities favours words of the same type. For example, we may expect 'Spain' and 'France' to be more similar than 'Spain' and 'Barcelona', when this metric is used, since 'Spain' and 'France' are associated with the neighbors 'Madrid' and 'Paris' which are similar, and which are related in a similar way to the target words. In our experiments, we also consider a variant in which the relation vectors are only used for selecting the neighbors. The similarity itself is then calculated as:

$$\mathit{sim}(w_1, w_2) = \cos(\mathbf{v}_{w_1} \oplus \mu\mathbf{v}_{n_1}, \mathbf{v}_{w_2} \oplus \mu\mathbf{v}_{n_2})$$

We evaluate the proposed similarity measure on four well-known benchmarking datasets for word representation learning. These are: (1) `rg65` (Rubenstein and Goodenough, 1965); (2) `wordsim` (Finkelstein et al., 2001); (3) `mc` (Miller and Charles, 1991); and (4) the English portion of `semeval17` (Camacho-Collados et al., 2017). We restrict our experiment to single words, and do not consider multiword expressions (e.g., named entities), as this would require a different approach for compositional meaning representation. We compare against a baseline model based on cosine similarity between the vectors of the target words (`cosine`). As for our proposed models, and observing the similarity measurement described above, we consider a 10-dimensional relation space, without (10rv$_w$) and with (10rv$_r$) the relation vector as part of the similarity computation. We also provide results stemming from using the original 1800-dimensional relation vector model. As is customary in the literature, we use Pearson's (**p**) and Spearman's (**s**) correlation coefficients as evaluation metrics, as well as their average (**avg.**). Table 3 shows that the 10rv$_w$ variant consistently outperforms the word-level baseline. Somewhat surprisingly, the variant 10rv$_r$ (which uses the relation vector also in the similarity computation) performs consistently worse than the variant 10rv$_w$. When using the original 1800-dimensional vectors, however, the situation is reversed, with 1800rv$_r$ outperforming 1800rv$_w$, and achieving the best results overall (with the exception of `mc`). These results clearly show that the relation vectors capture valuable information for measuring word similarity, although the information captured by the 10-dimensional vectors may in some cases be too abstract for this purpose.

	`rg`			`wordsim`			`mc`			`semeval17`		
	p	**s**	**avg.**	**p**	**s**	**avg.**	**p**	**s**	**avg.**	**p**	**s**	**avg.**
`cosine`	77.2	76.0	76.6	64.9	69.4	67.1	79.2	80.0	79.6	69.4	70.0	69.7
10rv$_w$	78.1	77.0	77.5	66.0	69.6	67.8	79.7	80.7	**80.2**	70.2	70.8	70.5
10rv$_r$	77.4	75.5	76.4	65.8	69.5	67.6	78.8	77.9	78.3	70.0	70.7	70.3
1800rv$_w$	79.5	80.6	**80.0**	67.4	69.8	68.6	79.4	79.0	79.2	71.4	71.8	71.6
1800rv$_r$	78.9	80.2	79.5	68.1	70.1	**69.1**	79.2	79.7	79.4	72.2	73.0	**72.6**

Table 3: Correlation results for different configurations of our proposed approach and a competitor baseline based on cosine similarity of word embeddings.

4.3 Text Classification

Semantic Vector Networks may be thought of as a natural way of enriching word-level semantic representations, which may in turn be useful for informing a neural architecture with relational (e.g., commonsense or lexical) knowledge. We will focus on two well known tasks, namely text categorization and sentiment analysis. Our goal is to examine the extent to which the performance of a vanilla neural network increases by being injected vector graph information as a complement to the information encoded in each individual word embedding. The strength of our proposal lies in the fact that this information comes exclusively from corpora, and thus the need to rely on often incomplete, costly and language dependent ontological or lexical resources is avoided.

As evaluation benchmarks we use three text categorization datasets, namely `20news` (Lang, 1995), `bbc` (Greene and Cunningham, 2006) and `reuters` (Lewis et al., 2004). We also consider two polarity detection datasets (positive or negative), namely the Polarity04 (`pol.04`) (Pang and Lee, 2004) and Polarity05 (`pol.05`) (Pang and Lee, 2005) datasets, and finally a 10k document subset of the *apps for android* (`apps4and.`) corpus[5] (He and McAuley, 2016), which features reviews and associated ratings on a scale from 1 to 5. The neural network model we use for our experiments is a combination of a CNN (LeCun et al., 1998) and a bidirectional LSTM (Hochreiter and Schmidhuber, 1997). CNNs have been evaluated extensively in text classification (Johnson and Zhang, 2015; Tang et al., 2015; Xiao and Cho, 2016; Conneau et al., 2017) and sentiment analysis (Kalchbrenner et al., 2014; Kim, 2014; dos Santos and Gatti, 2014; Yin et al., 2017), and this specific model (CNN+BLSTM) has been explored in different NLP benchmarks (Kim, 2014). Finally, as evaluation metrics we use precision (**p**), recall (**r**) and f-score (**f**), as well as accuracty (**acc.**).

To use SeVeN for text classification, we keep the exact same neural network architecture, but use enriched vector representations for each word. As a proof-of-principle, in this paper, this enriched vector representation is simply obtained by concatenating the word vector of that word, with vector representations of its top-10 neighbors according to PMI (ordered by this PMI score), together with the corresponding relation vectors. For example, with word embeddings of 300 dimensions and relation vectors of 10 dimensions, the input for each word is given by a 3400-dimensional vector. We list experimental

[5]Obtained from `http://jmcauley.ucsd.edu/data/amazon/`.

	bbc			20news			reuters-r56			apps4and.			pol.04	pol.05
	p	**r**	**f**	**p**	**r**	**f**	**p**	**r**	**f**	**p**	**r**	**f**	**acc.**	**acc.**
cosine	0.95	0.95	**0.95**	0.86	0.85	0.86	0.85	0.88	0.86	0.39	0.48	0.38	0.54	**0.78**
10rv	0.95	0.95	**0.95**	0.88	0.87	0.87	0.89	0.91	**0.90**	0.40	0.44	**0.40**	0.56	0.75
20rv	0.96	0.95	**0.95**	0.89	0.89	**0.89**	0.89	0.92	**0.90**	0.38	0.48	**0.40**	0.59	**0.78**
50rv	0.94	0.94	0.94	0.88	0.87	0.88	0.89	0.91	**0.90**	0.35	0.46	0.38	**0.60**	0.77

Table 4: Experimental results on six benchmarking datasets for text classification.

results for several configurations, where the number of neighbors stays fixed, but the relation vector (rv) changes in dimensionality (10, 20 or 50). Experimental results are provided in Table 4. We can see that for the 20-dimensional vectors, the results are consistently better (or at least as good as) the baseline. The results for the 10-dimensional and 50-dimensional vectors are similar, although these configurations perform slightly worse than the baseline for pol.05.

Overall, these results show the usefulness of the relation vectors and neighborhood structure, despite the rather naive way in which this information is used. It seems plausible to assume that performance may be further improved by using network architectures which exploit the graph structure in a more direct way.

5 Conclusions and Future Work

In this paper we have presented SeVeN, a dedicated vector space model for relational knowledge. These relation vectors encode corpus-based evidence capturing the different contexts in which a pair of words may occur. An initially high-dimensional relation vector is is further "purified" thanks to a simple *ad-hoc* autoencoder architecture, designed to only retain relational knowledge. We have explored the characteristics of these vector networks qualitatively, by showing highly correlated word pairs, as opposed to, for example, difference vectors. While the latter are often assumed to capture relational properties, we found that the relational similarities they capture largely reflect the similarities of the individual words, with little relational generalization capability. In addition, we have evaluated our SeVeN vectors in terms of their usefulness in two standard NLP tasks: word similarity and text classification. In both cases we obtained better results than baselines that use standard word vectors alone.

There are several interesting avenues for future work. First, an obvious way to improve these unsupervised representations would be to leverage structured knowledge retrieved from knowledge graphs and/or Open Information Extraction systems. Such knowledge could easily be exploited by feeding any available structured knowledge as additional inputs to the autoencoder. Another way in which structured knowledge could be harnessed would simply be to label relation vectors, i.e. to identify regions in the relation vector space that correspond to particular relation types (e.g. hypernymy). Another possibility would be to improve SeVeN by aggregating relation vectors along paths in the graph. In this way, we may learn to predict missing edges (or to smooth relation vectors that were learned from too few or too uninformative sentences), similarly to the random walk based strategies that have been developed for completing traditional semantic networks and knowledge graphs (Gardner et al., 2014).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by ERC Starting Grant 637277.

References

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert. 2018. Relation induction in word embeddings revisited. In *Proceedings of the 27th International Conference on Computational Linguistics*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Claudio Delli Bovi, Luis Espinosa Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In *Proceedings of EMNLP*, pages 726–736, Lisbon, Portugal, September. Association for Computational Linguistics.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proc. ACL*, pages 626–634.

Miao Fan, Kai Cao, Yifan He, and Ralph Grishman. 2015. Jointly embedding relations and mentions for knowledge population. In *Proc. RANLP*, pages 186–191.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 397–406.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, pages 377–384. ACM.

Víctor Gutiérrez-Basulto and Steven Schockaert. 2018. From knowledge graph embedding to ontology embedding: Region based representations of relational structures. *arXiv preprint arXiv:1805.10461.*

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proc. CoNLL*, pages 268–278.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 95–105.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2017. Modeling semantic relatedness using global relation vectors. *CoRR*, abs/1711.05294.

Richard Johansson and Luis Nieto Piña. 2015. Embedding a semantic network in a word space. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1428–1433.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. *NAACL.*

David Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 356–364.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 100–111.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

George A Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

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

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 6341–6350.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1351.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proc. HLT-NAACL*, pages 74–84.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1793–1803.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the mwe gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Robert Speer, Joshua Chin, and Catherine Havasi. 2016. ConceptNet 5.5: An open multilingual graph of general knowledge. *AAAI Conference on Artificial Intelligence*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1136–1141.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ivan Vulic and Nikola Mrksic. 2017. Specialising word vectors for lexical entailment. *CoRR*, abs/1710.06371.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *IJCAI*, pages 1390–1397.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the International Conference on Computational Linguistics*, pages 2335–2344.