A Retrofitting Model for Incorporating Semantic Relations into Word Embeddings

Sapan Shah[1,2], **Sreedhar Reddy**[1], **and Pushpak Bhattacharyya**[2]

[1]TCS Research, Tata Consultancy Services, Pune
[2]Indian Institute of Technology Bombay, Mumbai
{sapan.hs,sreedhar.reddy}@tcs.com
pb@cse.iitb.ac.in

Abstract

We present a novel retrofitting model that can leverage relational knowledge available in a knowledge resource to improve word embeddings. The knowledge is captured in terms of relation inequality constraints that compare similarity of related and unrelated entities in the context of an anchor entity. These constraints are used as training data to learn a non-linear transformation function that maps original word vectors to a vector space respecting these constraints. The transformation function is learned in a similarity metric learning setting using Triplet network architecture. We applied our model to synonymy, antonymy and hypernymy relations in WordNet and observed large gains in performance over original distributional models as well as other retrofitting approaches on word similarity task and significant overall improvement on lexical entailment detection task.

1 Introduction

Word embedding models (Pennington et al., 2014; Mikolov et al., 2013) are primarily inspired from the distributional hypothesis (Harris, 1954) viz. words that appear in similar context tend to have similar meaning. However, these models have one major drawback: they mix semantic similarity with other types of semantic relatedness (Hill et al., 2015). Consider for example, *cheap* and *expensive*. Though completely opposite in meaning, these words tend to occur in nearly identical contexts and end up having similar distributional vectors. This is problematic for many end applications such as sentiment analysis, text simplification, and so on. To address this issue, researchers have proposed various models to combine information from external knowledge sources such as WordNet, Freebase, etc. into unsupervised learning of word embeddings. These models mainly focus on the constraints extracted from various types of relations such as synonymy, antonymy, hypernymy, etc. At a high level, these models are categorized into: Joint specialization models (Yu and Dredze, 2014; Liu et al., 2015; Xu et al., 2014); and Retrofitting models (Faruqui et al., 2015; Wieting et al., 2015; Glavaš and Vulić, 2018; Kamath et al., 2019). Joint specialization models typically modify the optimization objective of distributional models by integrating the constraints into the objective function. Whereas, retrofitting models update the word vectors of distributional models in a post-processing training step using data generated from the constraints. Current retrofitting models have one limitation. They use constraints that tend to push cosine similarity to extremes (+1 or -1). While this works well for relations such as synonymy and anotonymy, it does not work so well for relations such as hypernymy, holonymy, etc. We need an approach that works for all relations while striking the right balance with distributional semantics.

In this work, we present a new method to obtain constraints from all types of relations present in a knowledge resource. The constraints are in the form of relation inequalities. The central idea is: if an entity ent_a is related to entity ent_b with relation type rel and is not related to entity ent_c by the same relation, then ent_a is semantically closer to ent_b than ent_c in the context of the relation rel. The

corresponding inequality can then be stated as: $sim_{rel}(ent_a, ent_b) > sim_{rel}(ent_a, ent_c)$. Using such relation inequality constraints has the following advantages: 1) They are not just limited to synonymy and antonymy relations but can also be generated from other lexical relations such as hypernymy, holonymy, and so on. 2) They can also be generated from any relation type including relations from non-lexical knowledge graphs such as FreeBase.

We use the generated inequality constraints as training data to learn a non-linear transformation function that maps original word vectors to a vector space respecting these constraints. The transformation function is learned in a similarity metric learning setting using Triplet network architecture (Wang et al., 2014; Schroff et al., 2015). We applied our model to synonymy, antonymy and hypernymy relations in WordNet and observed large gains in performance over retrofitting benchmarks on word similarity task and significant overall improvement on lexical entailment (LE) detection task. The main contributions of this work are: (1) A new method to obtain constraints from all relation types in a retrofitting setting with its demonstration on WordNet relations; (2) Using Triplet network based similarity metric learning for a softer, more balanced integration of constraints; (3) A detailed experiment on word similarity as well as LE detection tasks to show the effectiveness of the proposed approach.

2 Constraints from WordNet Relations

This section presents a set of rules to obtain relation inequality constraints from synonymy, antonymy and hypernymy relations in WordNet (Miller, 1995). Each rule involves a triplet of entities (v_a, v_p, v_n) in which we refer to v_a, v_p and v_n as the *anchor*, *positive* and *negative* entities respectively. A constraint is generated such that the anchor is closer to the positive than the negative entity, with a margin to indicate minimum separation. It can be set in the range of $[0, 2]$ corresponding to minimum versus maximum separation on cosine distance.

Similarity Relationship Constraints: We represent the set of unique words appearing in all synsets as nodes of a graph G. We then add a labeled edge *syn* between two nodes if the corresponding pair of words belongs to the same synset. The inequality constraints for $r = syn$ are then obtained using,

$$\forall v_a, v_p, v_n; (v_a, r, v_p) \in G; (v_a, r, v_n) \notin G; \quad sim(v_a, v_p) > sim(v_a, v_n) + margin_r \tag{1}$$

i.e. a pair of entities associated by a specific relation r are semantically closer than a pair of entities that are not in that relation. Specifically for $r = syn$, we sample negative words v_n using antonymy relation. For instance, consider a triplet *(bright, clever, stupid)*. With respect to anchor word *bright*, *clever* is closer than *stupid* as the former is a synonym whereas the latter is an antonym of *bright*. This generates the following constraint: $sim(bright, clever) > sim(bright, stupid) + margin_{syn}$

Type Hierarchy Constraints: In addition to the word nodes, we also create type nodes to represent synsets in G and add edges between them using hypernymy relation to capture type hierarchy (edge label: $subtype$). Moreover, we also add an edge between a word and the type corresponding to its synset (edge label: $type$). We then apply the following rule to generate type hierarchy constraints.

$$\begin{aligned} \forall v_a, v_p, v_n; \quad & (v_a, type, t_1), (v_p, type, t_2), (v_n, type, t_3), (t_1, subtype, t_2), (t_2, subtype, t_3) \in G; \\ & sim(v_a, v_p) > sim(v_a, v_n) + margin_{hier} \end{aligned} \tag{2}$$

i.e. a pair of entities closer in type hierarchy are semantically closer compared to a pair farther in type hierarchy. For instance, consider a triplet *(coach, railcar, vehicle)*. With respect to anchor word *coach*, *railcar* is closer than *vehicle* as the former is a direct hypernym of *coach* whereas the latter is an indirect hypernym through *railcar*. This generates the following constraint: $sim(coach, railcar) > sim(coach, vehicle) + margin_{hier}$.

It should be noted that rules 1 and 2 above can be used to obtain constraints from any knowledge graph. Rule 1 in its general form encodes any relation r where a triplet (v_a, v_p, v_n) is sampled such that v_a and v_p are in relationship r while v_a and v_n are not. Similarly rule 2 can obtain type hierarchy constraints.

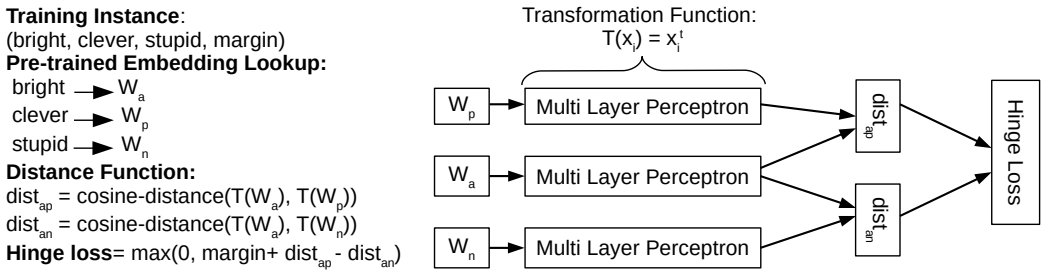

Figure 1: Retrofitting model for learning a non-linear transformation function

3 Transformation Function

We use the generated inequality constraints as training data (D) to learn a transformation function that maps pre-trained word embeddings to a vector space that respects these constraints. This function is learned using a Triplet network architecture in a similarity metric learning setting. The Triplet architecture (Hoffer and Ailon, 2015; Wang et al., 2014) provides a way to learn transformation from input space to representation space such that distances in the representation space approximates semantic distances in the input space.

Figure 1 shows the architecture and an example training instance to learn the transformation function. Let $X \in \mathbb{R}^{n \times d}$ represent the pre-trained embeddings for vocabulary of size n. For a training instance $(w_a, w_p, w_n, margin) \in D$, we first obtain corresponding pre-trained embeddings from X i.e. $(x_{w_a}, x_{w_p}, x_{w_n})$. These embeddings are then passed as input to a transformation function $T(x_i) = x_i^t$, a multi-layer feed forward neural network with weights W_T. Our model contains three identical copies of this network with shared parameters. These copies are then joined using a distance layer that computes two cosine distances viz. distance of the anchor word w_a from the positive word w_p and its distance from the negative word w_n i.e.

$$dist_{ap} = \text{cosine-distance}\big(T(x_{w_a}), T(x_{w_p})\big)$$
$$dist_{an} = \text{cosine-distance}\big(T(x_{w_a}), T(x_{w_n})\big)$$

These distances in the transformed vector space are then fed to a margin based hinge loss function. To reduce overfitting, we apply L_2 regularization on the weights W_T of the network. The loss function L_{hinge} used by our model is then,

$$L_{hinge} = \sum_{(w_a, w_p, w_n, margin) \in D} \Big(max(0, margin + dist_{ap} - dist_{an}) + \lambda_w \|(W_T)\|_2 \Big)$$

Similar to (Mrkšić et al., 2016; Glavaš and Vulić, 2018), we also include a regularization term L_{vsr} that penalizes vector space transformations that drastically change the topology of input vector space.

$$L_{vsr} = \sum_{(w_a, w_p, w_n, margin) \in D} \Big(\text{cosine-distance}(x_{w_a}, T(x_{w_a})) + \text{cosine-distance}(x_{w_p}, T(x_{w_p})) + \text{cosine-distance}(x_{w_n}, T(x_{w_n})) \Big)$$

The final loss function used by our model is then: $L = L_{hinge} + \lambda_{vsr} * L_{vsr}$ where λ_{vsr} is a hyper-parameter that determines how strictly the topology of original vector space is preserved. Once the network is trained, the learned transformation function $T(x_i)$ is applied to all the words in X to map the pre-trained word embeddings to a new transformed vector space $X^t \in \mathbb{R}^{n \times d}$.

4 Experimental Setup

To evaluate our retrofitting approach, we experimented with three pre-trained word embeddings that are learned using different distributional models: (1) GloVe (Pennington et al., 2014): trained on Common Crawl data; (2) Word2Vec (Mikolov et al., 2013): trained on Wikipedia dump available on polyglot project

(Al-Rfou' et al., 2013); (3) FastText (Bojanowski et al., 2017): trained on Wikipedia 2017. As explained in section 2, we use WordNet to obtain two types of constraints: (1) Similarity relationship constraints: a total of 425,732 constraints from synonymy and antonymy relations; (2) Type hierarchy constraints: a total of 100,100 constraints from hypernymy relation. The margin parameter is set to 0.6 and 0.2 for the similarity relationship constraints and the type hierarchy constraints respectively[1].

We compare our model (referred as **TripletNet** hereafter) with three state of the art retrofitting models: (1) **Counterfit** (Mrkšić et al., 2016): It defines the loss function as a weighted sum of terms that brings synonymous words closer, pushes antonymous words apart. However, it retrofits only those words that are present in the constraints (2) **ExplRetrofit** (Glavaš and Vulić, 2018): It retrofits vectors of all words in the vocabulary by learning a global specialization function using synonym and antonym constraints (3) **AuxGAN** (Ponti et al., 2018): It learns the global specialization function using a generative adversarial network architecture. We also compare our model with the joint specialization approach of Liu et al. (2015) that updates word2vec optimization objective (referred as **SWE**).

		SimLex-999			SimVerb-3500		
		GloVe	FastText	Word2Vec	GloVe	FastText	Word2Vec
Lexical Overlap	PreTrained	0.3738	0.4409	0.3625	0.2264	0.3558	0.2531
	Counterfit	0.6038	**0.5949**	**0.5869**	0.4468	**0.4725**	0.4505
	ExplRetrofit	0.6252	0.5331	0.5364	0.5362	0.4182	**0.5290**
	AuxGAN	**0.6317**	0.3618	0.5672	0.4875	0.2980	0.4589
	SWE	-	-	0.5017	-	-	0.4001
	TripletNet-Sim	0.6014	0.5149	0.5316	0.5055	0.4348	0.4615
	TripletNet-Type	0.4288	0.4337	0.3687	0.3257	0.3494	0.2687
	TripletNet	0.6139	0.5349	0.5386	**0.5525**	0.4404	0.4953
Lexical Disjoint	PreTrained	0.3738	0.4409	0.3625	0.2264	0.3558	0.2531
	Counterfit	0.3702	0.4381	0.3631	0.2257	0.3578	0.2561
	ExplRetrofit	0.5265	0.526	0.3905	0.3553	0.4042	0.2634
	AuxGAN	0.5630	0.3339	0.4704	0.4194	0.2541	0.3540
	SWE	-	-	0.4612	-	-	0.3620
	TripletNet-Sim	0.5725	0.5124	0.4986	**0.5105**	0.4199	0.4336
	TripletNet-Type	0.4301	0.4319	0.3674	0.3081	0.3402	0.2650
	TripletNet	**0.5742**	**0.5314**	**0.5026**	0.5025	**0.4311**	**0.4541**

Table 1: Spearman's correlation (ρ) scores of our model and other benchmarks for three distributional embeddings on two word similarity datasets: SimLex-999 and SimVerb-3500

4.1 Word Similarity Task

We evaluate our approach on two word similarity datasets: SimLex-999 (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016) using Spearman's rank correlation (ρ). Similar to Glavaš and Vulić (2018), we use two evaluation settings i.e. **Lexical disjoint:** To effectively evaluate the generalization capability of retrofitting approaches, all words appearing in the evaluation datasets are removed from the training set; **Lexical overlap:** In this setting, all words appearing in the evaluation datasets are retained in training set.

Table 1 shows the results of our experiments. In both the evaluation settings, our model performs substantially better than the baseline pre-trained embeddings and the joint specialization model (SWE). In lexical disjoint setting, the Counterfit model does not improve beyond baseline as all the words present in the evaluation set are excluded from training. The ExplRetrofit and AuxGAN models perform better than Counterfit as they retrofit vectors of all words using a global specialization function. Our model performs even better as it does not put hard constraints on cosine similarity values. Instead, these values are learned

[1]The margin parameter for the inequality constraints are varied as: $\{0, 0.6, 1, 1.5, 2\}$ for similarity relationship constraints; and $\{0.05, 0.1, 0.2\}$ for type hierarchy constraints. The performance of the GloVe retrofitted embeddings on SimLex-999 evaluation set (refer section 4.1) is used to tune this parameter.

such that the anchor words are relatively closer to positive words than negative words thereby striking the right balance between distributional and relational semantics. In *lexical overlap* setting, retrofitting models perform significantly better than the baseline. Since the Counterfit model directly updates input word vectors pushing cosine similarity of synonyms to 1 and antonyms to -1 and many of the words in the evaluation set are already included in the training set, it seems to be performing better overall.

We also performed ablation test on the type of constraints. The embeddings retrofitted only using the similarity relationship constraints (TripletNet-Sim) perform significantly better than other benchmarks. However, the embeddings retrofitted only using the type hierarchy constraints (TripletNet-Type) bring only marginal improvement over baseline. This is on expected lines as similarity constraints are more important for the word similarity task. Combining both constraints (TripletNet) brings further improvement suggesting positive interaction between the embeddings of words across both types of constraints.

4.2 Lexical Entailment (LE) Detection Task

LE detection is a classification task to identify if a given pair of words is in a lexical entailment relation such as hypernymy, causality, and so on. We evaluate our approach on four datasets: Baroni (Baroni et al., 2012), WBLESS (Weeds et al., 2014), Kotlerman (Kotlerman et al., 2010) and Turney (Turney and Mohammad, 2014). The dataset splits provided by Levy et al. (2015) are used for the experiments on Baroni, Kotlerman and Turney. Whereas for WBLESS, we randomly split the data into train (70%) and test (30%) set. Given a pair of words (x, y) as input, we first represent them as the concatenation of their word embeddings (i.e. $\vec{x} \oplus \vec{y}$) and then learn a logistic regression classifier. In addition to the models explained earlier, we also compare our model with **LEAR** (Vulić and Mrkšić, 2018) that uses vector norms to define an asymmetric distance metric in order to leverage LE relations during training.

	PreTrained	Counterfit	ExpRetrofit	AuxGAN	LEAR	TripletNet	LEAR-M
Baroni	0.7313	0.7282	0.7519	0.7649	0.7687	**0.8078**	0.903
WBLESS	0.9349	0.9058	0.9346	**0.9442**	0.9385	0.9269	0.8885
Kotlerman	0.7021	0.7264	0.7262	**0.7472**	0.7118	0.7407	0.599
Turney	0.6982	0.6888	0.6923	0.714	0.6371	**0.7357**	0.6765

Table 2: Accuracy scores of our model and other benchmarks on LE detection datasets (GloVe embeddings)

Table 2 reports accuracy for various models on LE detection task. Overall, approaches that learn specialization function perform better than other approaches. Our model performs even better on Baroni and Turney, while comparable to AuxGAN on WBLESS and Kotlerman. We also learnt a model (**LEAR-M**) based on the asymmetric distance metric to identify LE pairs. This model performed significantly better on Baroni. However, it did not perform well on other datasets.

5 Conclusions and Future work

We present a novel retrofitting model that first generates inequality constraints from relational knowledge present in a knowledge resource. These constraints are then used as training data to learn a non-linear transformation function in a similarity metric learning setting using Triplet network architecture. We applied our model to synonymy, antonymy, and hypernymy relations in WordNet and observed large gains in performance over original pretrained embeddings as well as other retrofitting benchmarks on word similarity task and significant overall improvement on LE detection task.

We are currently evaluating our model on extrinsic tasks such as sentiment analysis, NER, etc. We also plan to incorporate relational knowledge present in non-lexical knowledge resources such as Freebase to further improve word embeddings.

References

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France, April. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182, Austin, Texas, November. Association for Computational Linguistics.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45, Melbourne, Australia, July. Association for Computational Linguistics.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, December.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *ICLR*.

Aishwarya Kamath, Jonas Pfeiffer, Edoardo Maria Ponti, Goran Glavaš, and Ivan Vulić. 2019. Specializing distributional vectors of all words for lexical entailment. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 72–83, Florence, Italy, August. Association for Computational Linguistics.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Natarul Language Engineering*, 16(4):359389, October.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June. Association for Computational Linguistics.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. 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)*, pages 1501–1511, Beijing, China, July. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California, June. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Edoardo Maria Ponti, Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 282–293, Brussels, Belgium, October-November. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society.

Peter Turney and Saif Mohammad. 2014. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21, 01.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1134–1145, New Orleans, Louisiana, June. Association for Computational Linguistics.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1386–1393, Washington, DC, USA. IEEE Computer Society.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1219–1228, New York, NY, USA. ACM.

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