

Learning to Negate Adjectives with Bilinear Models

Anonymous EACL submission

Abstract

TODO

1 Introduction

Identifying antonyms in a vector space model is a notoriously challenging task, because the vectors for related words of opposite polarity, such as *hot* and *cold*, are close to each other in most commonly-used word embedding models. Previous work has explored training and retrofitting embeddings to push antonyms further apart than synonyms in the vector space (Pham et al., 2015; Nguyen et al., 2016; Mrkšić et al., 2016), and using unsupervised measures to distinguish antonymous word pairs from other binary lexical relations (Santus et al., 2015). However, none of these methods provide us with a negation function which helps us locate an acceptable antonym for a given word in the space. In this paper we address the task of learning a negation mapping, given an arbitrary set of word embeddings. This task is important for composing larger units of meaning, where a phrase such as *not cold* might need to be assigned a vector space representation.

We focus on negating adjectives, and treat negation as prediction of a one-best antonym. For example, given the expression *not talkative* and the vector *talkative*, the negation function should return a word from the set *quiet*, *taciturn*, *uncommunicative*, etc.

Semantically, pairs of antonyms share a domain—e.g. *temperature*, but differ in their value—e.g. *coldness* (Turney, 2012; Hermann et al., 2013). The challenge for negation is to learn which features need to change, and in what direc-

tion, to change the value while retaining the semantic domain of the word.

In this paper we exploit the semantic neighbourhood of each adjective as a stand-in for the domain. We hypothesize that the relevant features for negating, say, a temperature adjective, differ from those for an emotion adjective. Therefore, we learn a function that uses a vector representing the neutral semantic context of the input word. Moreover, a negation function must be (informally) involuntary, i.e. its own inverse, so it must detect the “direction” in which the input word is more extreme than the neutral vector. This approach is inspired by Kruszewski et al. (2016), who observe that nearest neighbours in a vector space are a good approximation for the alternatives that humans produce for negation.

For this task we introduce a bilinear relational neural network architecture which has proven successful in identifying image transformations in computer vision, we are able to learn a negation relevant function. Our model outperforms other variants on a multiple choice antonym selection task, and more importantly learns to produce a one-best antonym with high precision.

2 Relational Encoders

2.1 Relational Autoencoders: Background

Relational autoencoders (RAE), also known as gated autoencoders (GAE), have been used in computer vision to learn representations of transformations between images, such as rotation or translation (Memisevic and Hinton, 2007; Memisevic, 2012, 2013). RAEs are a type of *gated network*, in which multiplicative connections between two related inputs are used for prediction or representation learning. The “gating” of one image vector by another allows the feature detectors to concentrate on the correspondences between the

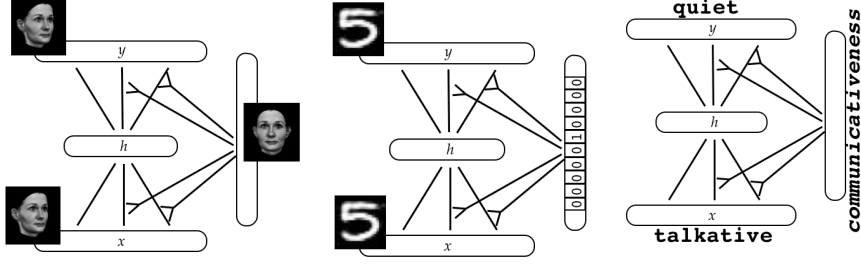


Figure 1: Neural network architectures for RAE, CCRAE, CCCRE.

related images, rather than being distracted by the differences between untransformed images (Figure 2). The key is the use of bilinear connections; there is a weight for every pair of units in the input vector and gate vector. For an overview of RAEs see Memisevic (2013); Sigaud et al. (2015).

We note that despite the terminology, the gates in an RAE perform a somewhat different function than in an LSTM (Hochreiter and Schmidhuber, 1997). Both network architectures use a nonlinearity to modulate the contents of a product; in an RAE this is an outer (bilinear) product while in an LSTM it is a Hadamard (element-wise) product. However, LSTM gates are memory gates which represent an internal hidden state of the network, while RAE gates are part of the input to the network.

An AE can be defined as follows (we omit bias terms for the sake of simplicity), where W_e are the encoder weights and W_d as the decoder weights. In autoencoders it is typical for them to be tied so that $W_d = W_e^T$.

$$\begin{aligned} h &= f(x) = \sigma(W_e x) \\ y &= g(h) = W_d h \end{aligned} \quad (1)$$

For an RAE, we have two inputs x and z . Instead of a weight matrix W we have a weight tensor $\overline{W} \in R^{n_H \times n_X \times n_Z}$. The RAE is defined as follows.

$$\begin{aligned} h &= f(x, z) = \sigma((\overline{W}_e z)x) \\ y &= g(h, z) = \sigma((\overline{W}_d h)z) \end{aligned} \quad (2)$$

Rudy and Taylor (2015) introduce a class-conditional gated autoencoder in which the gate is a one-hot class label, rather than a transformed version of the input image. For example, in the MNIST task the label represents the digit. This is like training an autoencoder per class but with weight sharing across classes. See Figure 2(b).

2.2 Continuous Class-Conditional Relational Encoders

Our bilinear model is a continuous class-conditional relational encoder (CCCRE). The model architecture is the same as an RAE with untied encoder and decoder weights. However, the training signal differs from a classic RAE in two ways. First, it is not an autoencoder, but simply an encoder, because it is not trained to reproduce the input but rather to transform the input to its antonym. Second, the encoder is class-conditional in the sense of Rudy and Taylor (2015). In a classic RAE the input and gate are related according to a transformation, but in a class-conditional AE the gate represents the class. Unlike the one-hot gates of Rudy and Taylor (2015), our gates, representing the semantic domain of the input vector, are real-valued. See Figure 2(c).

3 Experiments

3.1 Models

We compare the CCRE with a number of baseline models. The simplest is cosine similarity in the original vector space. We train a linear model (**Linear**) which maps the input word to its antonym, an Untied Encoder (**UE**) with a bottleneck hidden layer of 150 units, and a shallow feed-forward model (**FF**) with 600 hidden units rather than a bottleneck. To test whether the semantic domain is helpful in learning to negate, each of these models has a **Concat** version in which the input consists of the concatenated input word and gate. **UE-Concat** has 300 hidden units and **FF-Concat** has 600. We did not find using dropout to improve performance in any of our models.

3.2 Experimental Settings

We use publicly-available¹ 300-dimensional embeddings trained on part of the Google News dataset using the skip-gram with negative sampling model of Mikolov et al. (2013). Antonym training data was obtained from WordNet (hereafter WN), resulting in approximately 20K training pairs. We exclude any antonym pair with the input word in the test set.

Gates were obtained under three conditions. In the **standard** condition we collect all WN cohyponyms of an input word. If there are fewer than 10 WN cohyponyms, we increase the number to 10 with nearest neighbours from the original vector space. The gate is calculated as the vector centroid of the resulting word list.

In the **unsupervised** condition we do not use WN to find the semantic neighbourhood, but rather use the ten nearest neighbours from the original vector space. However, the target antonym is still supervised. In the **restricted** condition, we remove all cohyponyms of test input words from the training data. This is to test how important it is to have training examples with similar gates. For example, if *cold* is a test word, we remove *hot*, *cold*, *tepid*, *cool* etc. from the training data.

We used MSE loss. Hyperparameters were tuned on the GRE development set. The feed forward and CCCRE networks have hidden layers of 600 units, while the UAE has a hidden layer of 150, and 300 for UAE-Concat. Minibatch size was 48 for bilinear networks and 16 for the others. Number of epochs 400 for feed-forward, 200 for bilinear, 300 for UAE, 100 for linear. Optimization adadelata with $\rho = 0.95$.

3.3 Evaluation

We evaluated our models with two experiments. Experiment 1 uses the Graduate Record Examination (GRE) questions of Mohammad et al. (2013). The task is, given an input word, to pick the best antonym out of five options. An example from the development set is shown in (3), where the input word is *piquant* and the correct answer is *bland*. We restrict the questions to those in which both input and target words are adjectives.

piquant: (a) shocking (b) jovial (c) rigorous
(d) merry (e) **bland** (3)

¹<https://code.google.com/archive/p/word2vec/>

Method	Training Condition		
	Stand.	Unsup.	Restr.
Random	0.20	—	—
Cosine	0.50	—	—
Linear	0.56	0.56	0.53
Linear-Concat	0.66	0.59	0.63
UE	0.57	0.55	0.52
UE-Concat	0.63	0.58	0.61
FF	0.58	0.54	0.51
FF-Concat	0.65	0.56	0.63
CCCRE	0.69	0.60	0.65

Table 1: Accuracy on the 367 multiple-choice adjective questions in the GRE test set.

We evaluate by predicting the antonym for the input word. We measure cosine distance from the predicted vector to each of the five terms in the multiple choice question, and choose the closest term. We report accuracy, i.e. percentage of questions answered correctly.

Experiment 2 evaluates the precision of the models. Recall that our goal was to negate an adjective given its word vector. A natural criterion for success is whether the model returns a good antonym at rank 1, or a few good antonyms at rank 5, rather than returning any particular antonym as required by the GRE task.

We use two datasets: the GRE test set and a set of 99 adjectives and their antonyms from a dataset collected by Lenci and Benotto according to the guidelines of Schulte im Walde and Köper (2013). For each input word we retrieve the five nearest neighbours of the model prediction and check against a gold standard whether each neighbour is an acceptable antonym.

Gold standard antonyms for an input word w include w 's antonyms from the test sets and the WN antonyms of w . Following Gorman and Curran (2005), to minimise false negatives we improve the coverage of our gold standard by expanding it with antonyms from the online version of Roget's 21st Century Thesaurus, Third Edition.² We report precision at ranks 1 and 5.

4 Results and Discussion

Table 1 shows the results of Experiment 1. A random baseline results in 0.20 accuracy. The cosine similarity baseline is already fairly strong at 0.50,

²<http://thesaurus.com>

Method	GRE						Lenci					
	Stand.		Unsup.		Restr.		Stand.		Unsup.		Restr.	
	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5
Cosine	0.05	0.07	—	—	—	—	0.13	0.10	—	—	—	—
Linear	0.36	0.29	0.34	0.29	0.32	0.28	0.29	0.25	0.30	0.24	0.29	0.23
Linear-Concat	0.39	0.33	0.43	0.34	0.36	0.31	0.33	0.28	0.31	0.27	0.32	0.27
FF	0.37	0.32	0.34	0.30	0.08	0.15	0.30	0.24	0.27	0.23	0.22	0.19
FF-Concat	0.36	0.30	0.46	0.40	0.37	0.34	0.34	0.26	0.28	0.26	0.34	0.27
UE	0.38	0.33	0.36	0.32	0.37	0.31	0.28	0.22	0.27	0.23	0.23	0.20
UE-Concat	0.38	0.33	0.43	0.38	0.27	0.31	0.33	0.28	0.34	0.27	0.28	0.25
CCCRE	0.66	0.49	0.52	0.42	0.52	0.38	0.39	0.32	0.46	0.32	0.34	0.30

Table 2: Precision at ranks 1 and 5 on the GRE and Lenci datasets.

suggesting that in general about two out of the five choices are closely related to the input word.

Information about the semantic domain clearly provides useful information for this task, because the **Concat** versions of the Linear, UE, and FF models achieve several points higher than the models using only the input word. The linear model achieves a surprisingly high 0.66 accuracy under standard training conditions. The CCCRE model achieves the highest accuracy across all training conditions, and is the only model that beats the linear baseline, suggesting that bilinear connections are useful for antonym prediction.

All the models show a notable loss of accuracy in the **unsupervised** condition, suggesting that despite the findings of Kruszewski et al. (2016), the alternatives found in the vector neighbourhood are not as accurate for this task. Even in this setting, however, CCCRE achieves a respectable 0.60 accuracy.

In the **restricted** condition, all models without the gate vector perform near the cosine baseline, suggesting that in the standard setting they were simply memorising antonyms of semantically similar words. The models with gates retain a higher level of accuracy, suggesting that they are able to generalise across different semantic classes.

Although CCCRE achieves the highest accuracy in Experiment 1, the GRE task does not reflect our initial goal, namely to generate a one-best antonym. CCCRE sometimes fails to predict the specific antonym in the GRE dataset, but still makes a good prediction. For input word *doleful*, the model fails to choose the GRE target word *merry*, preferring instead *sociable*. However, the top three nearest neighbours for the predicted antonym of *doleful* are *joyful*, *joyous*, and *happy*, all very acceptable antonyms. Experiment 2 measures precision directly.

Table 2 shows the results of Experiment 2. In

the standard training condition with supervised gates, CCCRE shows an impressive P@1 of 0.66, i.e. two thirds of the time it is able to produce an antonym of the input word as the nearest neighbour of the prediction. All of the other models score less than 0.4. In the other training conditions CCCRE retains the ability to give a one-best antonym about half the time.

The Lenci dataset is more challenging [TODO: say why]. However, CCCRE still achieves the highest precision.

[TODO: try to fit an example of the top 5 for CCCRE and one other model for an example word or two!]

5 Related Work

Widdows (2003) introduces a binary negation function for vector spaces, e.g. *suit NOT lawsuit*, which is used for word sense disambiguation in Information Retrieval. Turney (2012); Hermann et al. (2013) propose a multi-part vector representation that separates domain and value, but this does not allow negation of a word in an arbitrary vector space. Pham et al. (2015); Nguyen et al. (2016); Mrkšić et al. (2016) use ontologies such as WN or domain-specific ontologies to retrofit embeddings, or learn embeddings with a secondary objective, pushing synonyms closer together and antonyms further apart. This improves the overall vector space and makes it more likely that close words will be synonyms, but does not provide a way to find antonyms in the space. Santus et al. (2014a,b, 2015) perform synonym-antonym discrimination using an unsupervised measure based on the relative salience of shared contexts. This type of relation classification does not make it possible to negate an arbitrary word. Mohammad et al. (2013) use a supervised thesaurus-based method augmented with co-occurrence statistics on the GRE task (including nouns and verbs).

[TODO: vision citations for a couple of applications of RAEs]

6 Conclusion

We have shown that a neutral context vector, representing the set of alternatives, improves antonym prediction in linear and non-linear models, and that the multiplicative connections in a bilinear model are effective at learning to negate adjectives. In the broader context of modelling closed-class function words, it is perhaps not too far-fetched to imagine that a specialised network with multiplicative connections might be necessary to model these kinds of words.

Our implementation is a simple one, using a full tensor for the bilinear layer, and takes several hours to train on a GPU. Future work will exploit the many options for reducing the number of parameters to be learned (Alain and Olivier, 2013) and address negation of nouns and verbs, where negation is more likely to involve a probability distribution over alternatives—e.g. *didn't run* might mean *walk* or *amble*—rather than predicting a single best antonym.

References

- Droniou Alain and Sigaud Olivier. 2013. Gated autoencoders with tied input weights. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA.
- James Gorman and James R. Curran. 2005. Approximate searching for distributional similarity. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 97–104, Ann Arbor.
- Karl Moritz Hermann, Edward Grefenstette, and Phil Blunsom. 2013. “not not bad” not “bad”: A distributional account of negation. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 74–82, Sofia, Bulgaria.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Germán Kruszewski, Denis Paperno, Raffaella Bernardi, and Marco Baroni. 2016. There is no logical negation here, but there are alternatives: Modeling conversational negation with distributional semantics. *Computational Linguistics*, 42:xxx–xxx.
- Roland Memisevic. 2012. On multi-view feature learning. In *Proceedings of ICML*.
- Roland Memisevic. 2013. Learning to relate images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1829–1846.
- Roland Memisevic and Geoffrey Hinton. 2007. Unsupervised learning of image transformations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.
- Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2013. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.
- Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL*, pages 142–148, San Diego.
- Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonymsynonym distinction. In *Proceedings of ACL*, pages 454–459, Berlin.
- N. Pham, A. Lazaridou, and M. Baroni. 2015. A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of ACL*, pages 21–26.
- Jan Rudy and Graham Taylor. 2015. Generative class-conditional denoising autoencoders. In *Proceedings of ICLR Workshop*.
- Enrico Santus, Qin Lu, Alessandro Lenci, and Chu-Ren Huang. 2014a. Unsupervised antonym-synonym discrimination in vector space. In *Atti della Conferenza di Linguistica Computazionale Italiana (CLIC-IT 2014)*, Pisa, Italy.
- Enrico Santus, Qin Lu, Alessandro Lenci, and Chu-Ren Huang. 2014b. Taking antonymy mask off in vector space. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation (PACLIC 2014)*, Phuket, Thailand.
- Enrico Santus, Alessandro Lenci, Qin Lu, and Chu-Ren Huang. 2015. When similarity becomes opposition: Synonyms and antonyms discrimination in DSMs. *Italian Journal of Computational Linguistics*, 1(1):41–54.

500	Sabine Schulte im Walde and Maximilian Köper.	550
501	Pattern-based distinction of paradigmatic rela-	551
502	tions for German nouns, verbs, adjectives. In	552
503	<i>Language Processing and Knowledge in the</i>	553
504	<i>Web</i> , pages 184–198. 2013.	554
505	Olivier Sigaud, Clément Masson, David Filliat,	555
506	and Freek Stulp. 2015. Gated networks: an	556
507	inventory. arXiv:1512.03201 [cs.LG].	557
508	Peter D. Turney. 2012. Domain and function:	558
509	A dual-space model of semantic relations and	559
510	compositions. <i>Journal of Artificial Intelligence</i>	560
511	<i>Research</i> , 44:533–585.	561
512	Dominic Widdows. 2003. Orthogonal negation	562
513	in vector spaces for modelling word-meanings	563
514	and document retrieval. In <i>Proceedings of ACL</i> ,	564
515	pages 136–143, Sapporo, Japan.	565
516		566
517		567
518		568
519		569
520		570
521		571
522		572
523		573
524		574
525		575
526		576
527		577
528		578
529		579
530		580
531		581
532		582
533		583
534		584
535		585
536		586
537		587
538		588
539		589
540		590
541		591
542		592
543		593
544		594
545		595
546		596
547		597
548		598
549		599