Is there a linear subspace in which the difference vectors of word analogy pairs are parallel?*

Stephen Taylor[0000−0001−6702−7900] and Tomáš Brychcin

[1] University of Western Bohemia, Pilsen, Czech Republic
stepheneugenetaylor@gmail.com
[2] Brychcin@kiv.zcu.cz

Abstract. Since Mikolov introduced vector arithmetic computation of word analogies, they have inspired both enthusiasm and disdain. If the arithmetic computation works, the relationship encoded in the word vectors should manifest itself as parallel difference vectors, and if the difference vectors are parallel, this should appear in two-dimensional projections. For Principal Component Analysis (PCA) computed on a handful of relations, this seems to be true, perhaps with some variation which might explained by multiple word senses, or imprecision in the trained position of words in the vector space. However, PCA on the entire vocabulary typically shows a wide range of directions for difference vectors in the same relation.

We suggest that there is a subspace for each relation, in which the difference vectors are parallel. This corresponds to the intuition that only a subset of the senses for each word participates in the relation. To approximate such a subspace, we train a linear transformation which moves a portion of the pairs in a relation so that the difference vectors are nearly parallel to each other, while minimizing the movement of witness words, and we see that there is a net improvement in evaluating not only analogies which include pairs in the training set, but also analogies between held-out pairs in the same relation. The trained transformation thus seems to isolate semantic composition expressed by the relation.

1 Introduction

Mikolov et al. [11] introduced solving word analogies with vector arithmetic in the same paper that introduced **word2vec**, the algorithms and software release for rapidly constructing CBOW and Skip-gram semantic vector spaces. Word analogies fired the imagination, because they are similar to some standard questions on human intelligence tests [16]. Briefly, Mikolov [11] asserted that if $\mathbf{s}(word)$ is the vector for $word$, then the word analogy *Man : King :: Woman : ?* can be solved for *?* = *Queen* by finding

* This work has been supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

the word closest in the vector space to $\mathbf{s}(King) - \mathbf{s}(man) + \mathbf{s}(woman)$. They provided a corpus of word analogies which test this idea, now called the Google word analogy corpus, which we use in this paper. This computation scheme is equivalent to the claim that in the semantic vector space, vector addition is equivalent to semantic combination, a claim made later that same year by Mikolov [12].

In Figure 1, we show four different PCA projections of the difference vectors for the `capital-common-countries` relation from a different vector space. The difference vectors are all aligned to start from the origin, which brings out the parallelism, or lack of it, sharply. Because each figure has a different set of basis vectors, each rosette is aligned and centered differently. The difference vectors are closest to parallel in Figure 1B, in which the eigenvectors, or axes of the projection, are computed based only on the 23 distinct pairs, or 46 words, in the relation. In general, the difference vectors in analogy files (hereafter *relations*) are not parallel in 300 dimensions [8,9] but can look more aligned after projection to two dimensions, particularly if the PCA is done on a subset of the data.

If their difference vectors were actually parallel and of the same length, then word analogies would have 100% evaluation success, which would make them formidable tools, instead of the interesting demonstrations which they now are.

Although none of the PCA projections captures much of the variation, they make it seem plausible that there might be a linear transformation of the vector space which minimizes information irrelevant to a particular relation.

We set ourselves the goal of finding, for each relation, a linear transformation such that the difference vectors in the transformed space would be parallel.

2 Related Work

Coecke et al. [1] proposed building an algebra for composing word vectors into phrases.

Mitchell and Lapata [14] consider how vectors in a semantic space, in their case Latent Semantic Analysis (LSA) vectors describing usage context for individual words, might be composed into phrases. They asked human beings to rate phrase similarity for two-word phrases, and attempted to correlate the human similarity scores with scores computed using nine different *composition functions* applied to the LSA vectors of the individual words. They found the best correlation for element-wise multiplication. Vector addition was in the middle of the group.

Levy et al. [6] were early investigators of vector-space word analogies; their article includes two versions of the [13] method, and a new scheme of their own. They exhibited a very-high dimension word embedding, in which context counts are dimensions, to explain how vector addition could plausibly perform semantic composition. They describe all the algorithms as forms of vector similarity calculation.

In [7] Levy examines various examples of hypernym-analogies, and concludes that what they measure is not whether the analogy is valid, but whether or not a word is a hypernym of *something*.

Linzen [8] argues that frequently the difference vector is irrelevent in analogy evaluation, and that a nearest neighbor effect overwhelms it.

Drozd et al. [2] notes that averaging the difference vector over many pairs makes it more reliable, and that then using logistic regression to determine a class for second words in a pair for the relation and checking whether a proposed answer is in the class gives excellent results. They tested their algorithm using two held-out pairs in the relation. This work has more emphasis on 'solving' analogies than ours, and less on the question of whether difference vectors are 'real'.

Finley et al. [3] test whether the vector-arithmetic strategy (without exclusion) is an improvement on word similarity on several test-sets. They concluded that some relations don't work well, but several do; They categorize the most successful analogy types as *Inflectional Morphology*; *Named Entities*; *Gendered Nouns*. Least successful types are *Derivational Morphology* and *Lexical Semantics*.

Gittens et al. [4] consider semantic combination, as a mathematical operation on word vectors in a syntactic space generated with the Skip-gram algorithm. They provide a model for paraphrases in terms of the target and context vectors used in Skip-Gram algorithm, which leads to a formula for finding them, and conclude that when Skip-gram is applied to a corpus with a uniform word distribution, semantic combination is vector addition. Natural language words follow a Zipf-like distribution, but Finley [3] note that analogies in which all four words have similar frequency seem to work better.

Vylomova et al. [17] train linear kernel SVMs to determine from their difference vectors whether pairs were members of a relation or not. They find using negative samples in the training set improves precision but lowers recall.

Konkol et. al. [5] trained a linear transform from place-names to geographical coordinates, demonstrating that vector semantic spaces apparently have some location information encoded in them. Szymanski [15] got a similar result with a different method, by training word vectors with parallel corpora from specific time periods in order to look for equivalents such as Ronald Reagan in 1987 is like Bill Clinton in 1997.

3 Methodology

We downloaded a semantic vector space file[3] and edited it to keep only the first 150,000 words. The vectors for this file have 300 32-bit floating point elements. We edited the Google analogy dataset[4] to break the analogies into pairs, which lets us consider training- and test-sets from among the pairs in each analogy type, which we then call a *relation*.

The fourteen relations of the Google analogy set, and a few statistics about each are shown in Table 1.

Some of the relations have fewer pairs than appear in the downloaded testset; this is because pairs with out-of-vocabulary words are eliminated. This occurs because we limited the size of the vocabulary to 150 thousand words. These are the most frequent words in the Google news corpus from which the word vector space was built. The downloaded file has vectors for about 3 million word-forms, but searching through so

[3] `GoogleNews-vectors-negative300.bin.gz` from https://code.google.com/archive/p/word2vec/

[4] http://download.tensorflow.org/data/questions-words.txt

relation	analogies	pairs	base correct	base spares	base accuracy
capital-common-countries	506	23	179	236	0.35
capital-world	1482	39	380	753	0.26
city-in-state	1560	40	304	942	0.19
country-currency	342	19	45	109	0.13
family	506	23	177	253	0.35
gram1-adjective-adverb	992	32	17	271	0.02
gram2-opposite	756	28	15	300	0.02
gram3-comparative	1332	37	501	687	0.38
gram4-superlative	1122	34	234	699	0.21
gram5-present-participle	1056	33	64	771	0.06
gram6-nationality-adjective	1482	39	1166	203	0.79
gram7-past-tense	1560	40	161	871	0.10
gram8-plural	1332	37	67	1153	0.05
gram9-plural-verbs	870	30	136	462	0.16

Table 1. Some statistics on the Google word analogies

many vectors to find a near-match takes 20 times as long as searching through the smaller list.

The **analogies** column in Table 1 is computed from the number of pairs as

$$\text{analogies} = \text{pairs}(\text{pairs} - 1) \tag{1}$$

Thus for any two different pairs, we can make two analogies. For some analogy types we could make four by reversing the pairs, but we do not do this. For example the analogy (from the `gram8-plural` relation)

$$mouse : mice :: cat : cats \tag{2}$$

could be manipulated to put any of the four words in the final position that the algorithm calculates. However, this is not obviously true for the analogy (from the `city-in-state` relation)

$$Fresno : California :: Tucson : Arizona \tag{3}$$

because if we provide the state, there seem to be many equally good answers for a city in the state.

The **base correct** column is the number of analogies for which vector addition provides the expected answer as the first choice. Mikolov's version of the algorithm excludes the three other words in the analogy from being considered. We provide the column **spares** to record when the correct answer would be provided by this work-around. Although word similarity is interesting, it has nothing to do with phrase composition by vector addition. A glance at the table shows that spares are a major contributor to the usual statistics for these relations.

3.1 Transforming the Semantic Space

We define the semantic vector space $\mathbf{S}$ as a matrix, each row of which is a semantic word vector $\mathbf{s}(w)$. The rows (word vectors) have a fixed order; in our case they are partially ordered by frequency of words in the corpus.

We denote pairs in a relation as p_i, and the two ordered elements of the pair as p_i^0 and p_i^1.

Our hypothesis is that for each different relation there is a non-zero matrix $\mathbf{C}$, such that for any two pairs in a relation, for example (from the `gram3-comparative` relation) $p_i = (\text{good}, \text{better}), p_j = (\text{bad}, \text{worse})$:

$$\left(\mathbf{s}(p_i^1) - \mathbf{s}(p_i^0)\right) \times \mathbf{C} = \left(\mathbf{s}(p_j^1) - \mathbf{s}(p_j^0)\right) \times \mathbf{C} \tag{4}$$

We are working with 300-dimensional vectors, and we have from 19 to 40 pairs in each relation. We need to hold out at least one pair for testing; we experimented with holding out 2,4,6,8,10,14, and 16 pairs. The rest of the pairs in the relation are the training set. We want to train the values in C so that the difference vectors between the words in the pairs in the training set are transformed by the matrix multiplication into points at the ends of vectors parallel to and equal to the mean difference vector for the set.

Before training we place the difference vectors for the pairs in the training set in a matrix $\mathbf{D}$ and their (identical) targets in a target matrix T.

We experimented with using both difference vectors and word-vectors with modified locations for training, but difference vectors give better results.

The target matrix $\mathbf{T}$ has rows $\mathbf{a}$ and $\mathbf{b}$ corresponding to the $\mathbf{s}(a)$ and $\mathbf{s}(b)$ rows in the training matrix $\mathbf{D}$.

The problem with this, is that a perfectly good solution could project all of S along the relation mean, densely packing the vector with points irrelevent to the relation. Ideally, word points in the vector space for words not in the relation would either stay put, or shift without condensing. To encourage this to happen, we tried a strategy of *pinning* difference vectors.

We add *pinned* vectors to the $\mathbf{D}$ and $\mathbf{T}$ matrices. These are a number of difference vectors between words chosen at random, which we want not to change; that is, we assume that a randomly chosen word pair (w_i, w_j) is not a pair of the relation, or is a *negative example.* For these vectors, the rows in $\mathbf{D}$ and $\mathbf{T}$ are the same, $\mathbf{s}(w_i)$.

One of the nice benefits of training difference vectors over the individual words of a pair, is that we can be more confident that two randomly chosen words are not part of the relation we are training for. In the case of syntactic relations like verb-past-tense, a randomly chosen word is quite likely to be a verb, and potentially part of the relation, even though it might not be part of the provided examples.

We train the matrix C with gradient descent, a learning rate of 0.01, for 3000 iterations, with a regularization constant of 0.99. Then, after training,

$$\mathbf{T} \approx \mathbf{D} \times \mathbf{C} \tag{5}$$

A final part of the training goal is a regularization step.

We train the vector space by gradient descent. The objective function consists of two terms:

1. The sum of squares of each coordinate in $(\mathbf{T} - \mathbf{D} \times \mathbf{C})$
2. A regularization term ρ. Considered as part of the objective function, this relates to a constant times the sum of the squared elements of the $\mathbf{C}$ array.

However, all we need during training is the partial derivatives with respect to the $\mathbf{C}$ array, which we compute as:

1. $\delta_1 = (\text{LearningRate})(2 * (\mathbf{D} \times \mathbf{C}) - \mathbf{T})^\top) \times \mathbf{D}$
2. $\rho \leq 1$ a fixed fraction for regularization

and apply at each training step as

$$\mathbf{C}' = (\rho)\mathbf{C} - \delta_1 \tag{6}$$

3.2 Evaluating the Analogies

In previous work with analogies, we have used two different kinds of normalization, in which every vector in the space $\mathbf{S}$ is changed.

Zero-centering first computes the mean of all vectors in the space, then subtracts the mean from every vector. As a result, the new mean of each vector coordinate is zero. This translation does not affect the angles between individual points in the space, but angles with the new origin are of course different than angles with the old origin were, and points which were on a single line extending from the origin no longer are.

Unit normalization computes the square root of the sum of the squares of the coordinates of a vector, that is, the Euclidean distance to the origin, and divides all the coordinates by this number, effectivly moving all points to the surface of a 300-dimensional hyper-sphere at unit distance from the origin. This transformation does not effect cosine distance, and it enables use to compute the cosine distance without recomputing vector norms, but it does change difference vectors. The vectors between points on the surface of the sphere now point off into empty space when applied to other starting points. Of course, for sufficiently short vectors, they might not point *far* off the surface; but the angle from a word to its nearest neighbor is typically near $\pi/4$, and this effectively shortens the vector.

In spite of these concerns, using these normalizations has apparently helped the success rate of analogy evaluation in the past. It has been suggested that just as unit normalization of vectors in the Information Retrieval vector space model [10] compensates for long documents, unit normalization of word vectors compensates for word frequency. But since in this study we are specifically concerned with difference vectors, and difference vectors are impacted by normalization (and in previous studies difference vectors are not the main factor in accuracy evaluations, as may be seen in Table 1) we elected not to normalize for these experiments.

Furthermore, Mikolov's word-search policy for finding the last word in the analogy explicitly rejects returning any of the first three words. This disallows some kinds of analogies, for example

$$\text{Prince Harry : Queen Elizabeth :: Prince William : ?} \tag{7}$$

where the answer is one of the guiding words, but more importantly, Linzen [8] points out that if the right answer is closer to the guide words than the computed vector, the computation and the difference vector are irrelevant, and only word similarity is being tested.

So we don't eliminate the other words in the analogy from consideration, and we consider the answer wrong if the nearest neighbor of the computed vector turns out to be one of the other words in the analogy, even if the correct answer is closer than any other word except for the guide words. We do keep track of this situation, primarily to able to compare the baseline figures with those of other researchers. We call this situation a *spare*, after the play in bowling in which some pins remain after the first ball, but are all knocked down by the second ball.

4 Experimental Results

The bar charts in Figure 2 and Figure 3 illustrate that the transformation improves performance on the word analogies for all of the four possible choices of pairs chosen from the training set and the held-out, untrained pairs. This suggests that the transformation might actually accomplish the goal of isolating the semantic component of the relation that it is trained on, including on words on which it is not trained.

Because the relations are small, and the chart is drawn from a particular training run, we see quite dramatic small number effects; the base accuracies between the four groups differ by large percentages, as do the trained accuracies.

The `capital-common-countries` relation is one of the best-performing of the Google set, but our statistics are lower than others, because we don't give credit for *spares*. We have a base performance before training of 179 successes for 506 analogies, or 0.35. We also note 236 spares, which would give a total accuracy of 0.82, but the spares are a measure of word-similarity, not of success in the difference vector calculation.

Table 2 shows experiments working out the improvements on the held-out set for various parameter values, in this case for pinned words instead of difference vectors.

Both Figure 2 and Figure 3 use 8 pairs held out and 50 pinned difference vectors.

	0	10	50	100	200	500
4	0.00	0.00	0.08	0.08	0.08	0.00
6	0.07	0.10	0.13	0.13	0.13	0.03
8	0.02	0.07	0.14	0.16	0.09	0.05
10	0.01	0.09	0.10	0.13	0.10	0.04
12	0.00	0.05	0.12	0.10	0.07	0.04
14	-0.04	0.02	0.09	0.11	0.08	0.03
16	-0.05	0.03	0.11	0.13	0.09	0.05

Table 2. Improvements in analogy performance between untrained pairs in the `capital-common-countries` relation after training.

The `capital-common-countries` relation has 23 pairs, and the held-out row labels at the left of the table show how many pairs were held out for testing. In the top row, four pairs are held out, so nineteen were used for training, and in the bottom row only seven pairs are available for training.

The column headings show the number of pinned points, or negative examples, that are used in training. The best run in this table shows an absolute improvement of 0.16 in the fraction of correct results, while the surface plot shows the total accuracy.

Relation	pairs/ ana- logies	base accu- racy	base accu- racy with spares	trained accu- racy	tp::tp		tp::hp		hp::tp		hp::hp	
					base	trnd	base	trnd	base	trnd	base	trnd
capital-common-countries	23/ 506	35%	82%	89%	23%	98%	34%	90%	35%	91%	40%	82%
capital-world	39 / 1482	25%	76%	97%	25%	99%	30%	89%	23%	91%	25%	66%
city-in-state	40 / 1560	19%	79%	96%	20%	98%	10%	88%	18%	95%	0	75%
country-currency	19 / 342	13%	45%	64%	13%	92%	12%	44%	16%	48%	3%	13%
family	23 / 506	34%	84%	71%	56%	99%	31%	70%	34%	74%	20%	46%
gram1-adjective-adverb	32 / 992	1%	29%	71%	0	91%	2%	49%	2%	50%	5%	19%
gram2-opposite	28 / 756	1%	41%	84%	1%	95%	3%	53%	2%	83%	6%	50%
gram3-comparative	37 / 1332	37%	89%	97%	42%	98%	31%	92%	23%	97%	13%	93%
gram4-superlative	34 / 1122	20%	83%	92%	17%	99%	24%	88%	22%	92%	27%	58%
gram5-present-participle	33/ 1056	6%	79%	83%	6%	99%	5%	68%	5%	78%	8%	48%
gram6-nationality-adjective	39/ 1482	78%	92%	96%	80%	96%	73%	97%	80%	96%	79%	95%
gram7-past-tense	40 / 1560	10%	66%	85%	10%	95%	9%	62%	11%	87%	11%	53%
gram8-plural	37 / 1332	5%	91%	97%	5%	99%	1%	84%	9%	97%	0	75%
gram9-plural-verbs	30 / 870	15%	68%	81%	18%	98%	9%	71%	20%	84%	10%	52%

Table 3. Some selected results for the google relations. tp=trained pair; hp = held-out pair

5 Discussion and Further Work

We have found a relation-specific linear transformation which improves the evaluation of word-analogies with vector arithmetic, including those pairs in the same relation which were held out of the training set. We believe that we are the first to consider this particular approach.

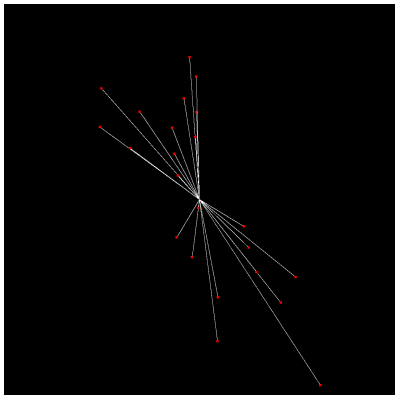
A number of the parameters of this experiment were chosen arbitrarily. For example, refraining from normalization, while plausible, may not be necessary (and unit-normalizing word vectors speeds up search.) Other strategies to keep the vector space from condensing may work better than negative examples.

In further work, we will explore the parameter space more fully.

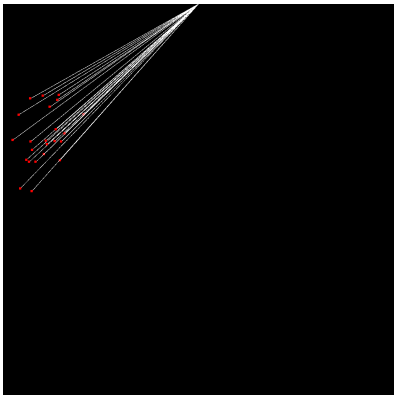
References

1. Coecke, B., Sadrzadeh, M., Clark, S.: Mathematical foundations for a compositional distributional model of meaning. Lambek Festschrit, special issue of Linguistic Analysis **36**, 345–384 (2010), http://arxiv.org/abs/1003.4394
2. Drozd, A., Gladkova, A., Matsuoka, S.: Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In: COLING (2016), https:/www.aclweb.org/anthology/C16-1332

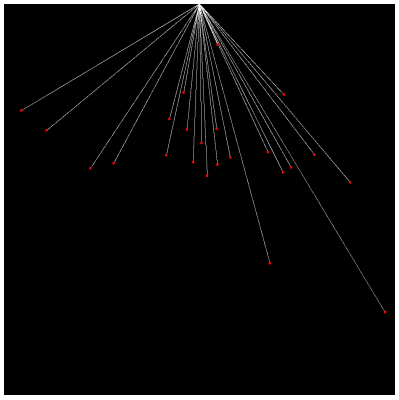
3. Finley, G.P., Farmer, S., Pakhomov, S.V.: What analogies reveal about word vectors and their compositionality. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (2017), http://www.aclweb.org/anthology/S17-1001
4. Gittens, A., Achlioptas, D., Mahoney, M.W.: Skip-gram – zipf + uniform = vector additivity. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. pp. pages 69–76 (2017), http://aclweb.org/anthology/P17-1007
5. Konkol, M., Brychcín, T., Nykl, M., Hercig, T.: Geographical evaluation of word embeddings. In: Proceedings of the The 8th International Joint Conference on Natural Language Processing. pp. 224–232 (2017), http://www.aclweb.org/anthology/I17-1023
6. Levy, O., Goldberg, Y.: Linguistic regularities in sparse and explicit word representations. In: Proceedings of the Eighteenth Conference on Computational Language Learning. pp. 171–180 (June 26-27 2014), http://www.aclweb.org/anthology/W14-1618
7. Levy, O., Remusu, S., Biemann, C., Dagan, I.: Do supervised distributional methods really learn lexical inference relations? In: North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT 2015) (2015), http://www.aclweb.org/anthology/N15-1098
8. Linzen, T.: Issues in evaluating semantic spaces using word analogies. In: RepEval@ACL (2016)
9. Liu, S., Bremer, P.T., Thiagarajan, J.J., Srikumar, V., Wang, B., Livnat, Y., Pascucci, V.: Visual exploration of semantic relationships in neural word embeddings. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS **24**(1), 553–562 (JANUARY 2018), https://ieeexplore.ieee.org/iel7/2945/8165924/08019864.pdf
10. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008), https://nlp.stanford.edu/IR-book/
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of workshop at ICLR. arXiv (2013), https://arxiv.org/pdf/1301.3781.pdf
12. Mikolov, T., Sutskever, I., Chen, K., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (2013), https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
13. Mikolov, T., tau Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: HLT-NAACL. vol. 13, p. 746–751 (2013), http://research.microsoft.com/pubs/189726/rvecs.pdf
14. Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cognitive Science **34**, 1388–1429 (2010), https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1551-6709.2010.01106.x
15. Szymanski, T.: Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers). pp. 448–453 (2017), http://www.aclweb.org/anthology/P17-2071
16. Turney, P.D., Littman, M.L.: Corpus-based learning of analogies and semantic relations. Machine Learning **60**(1), 251–278 (Sep 2005). https://doi.org/10.1007/s10994-005-0913-1, http://cogprints.org/4518/1/NRC-48273.pdf
17. Vylomova, E., Rimell, L., Cohn, T., Baldwin, T.: 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. pp. 1671–1682 (2016), http://aclweb.org/anthology/P16-1158



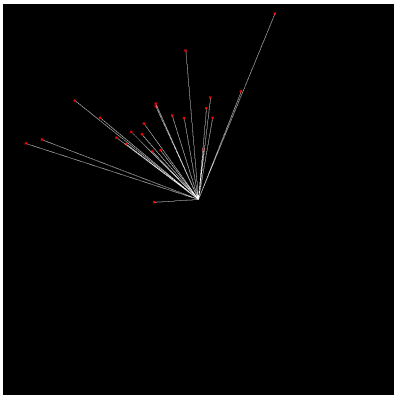
A. PCA uses whole vocabulary



B. PCA uses only relation file



C. PCA uses fourteen relation files, 936 words



D. PCA uses 467 words

Fig. 1. Various 2-D projections of `capital-common-countries`, using PCA on different sets of vectors.

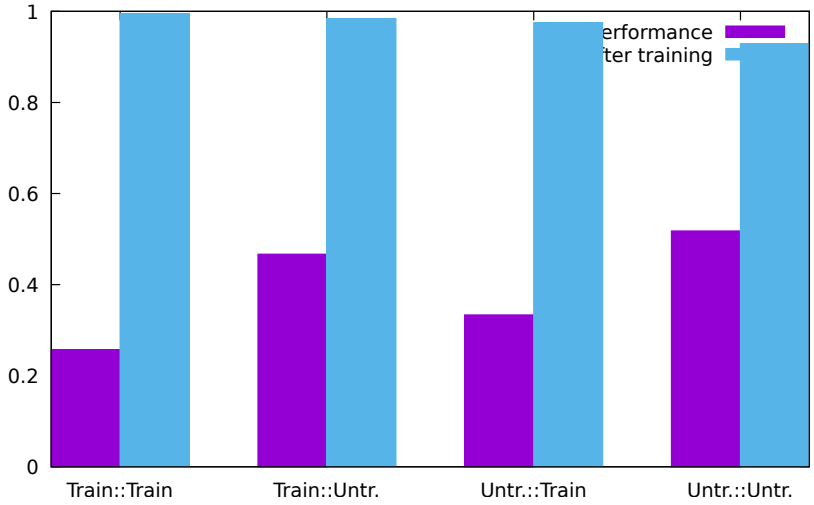


Fig. 2. Accuracy for capital-common-countries relation before and after trained transformation

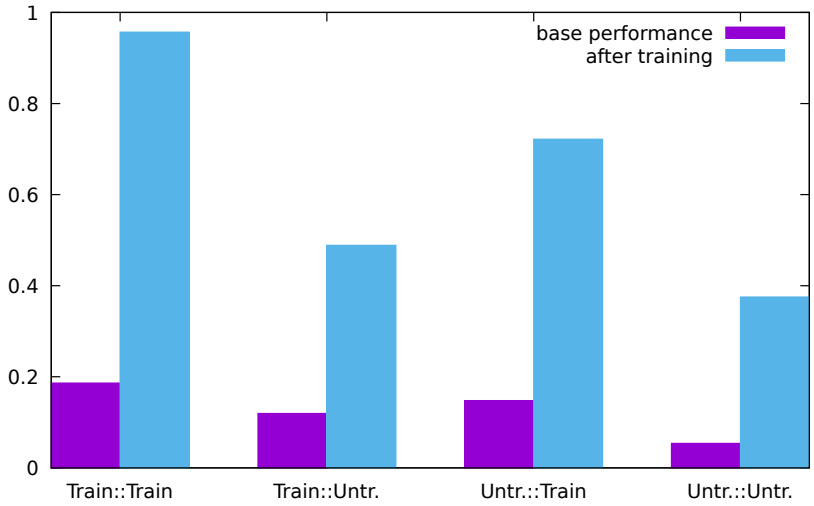


Fig. 3. Accuracy for `gram9-plural-verbs` relation before and after trained transformation