Combining Distributed Vector Representations for Words

Justin Garten and **Kenji Sagae** and **Volkan Ustun** and **Morteza Dehghani**
University of Southern California
Los Angeles, CA 90089, USA
{jgarten, mdehghan}@usc.edu, and {sagae, ustun}@ict.usc.edu

Abstract

Recent interest in distributed vector representations for words has resulted in an increased diversity of approaches, each with strengths and weaknesses. We demonstrate how diverse vector representations may be inexpensively composed into hybrid representations, effectively leveraging strengths of individual components, as evidenced by substantial improvements on a standard word analogy task. We further compare these results over different sizes of training sets and find these advantages are more pronounced when training data is limited. Finally, we explore the relative impacts of the differences in the learning methods themselves and the size of the contexts they access.

1 Introduction

Distributed vector representations allow words to be represented in a continuous space. By learning these representations using unsupervised methods over large corpora, these models capture key distributional aspects of word function and meaning. In particular, such representations provide a valuable response to issues of data sparsity by providing simple similarity measures between terms. Whether used indirectly in terms of those similarity measures (e.g. for smoothing in language models) or directly as features to a model for tasks such as parsing (Lei et al., 2014), these representations have proved increasingly valuable to a variety of NLP tasks (Bengio et al., 2013).

Given these benefits, a number of approaches have been explored for generating these representations beginning with early work in connectionist modeling (McClelland et al., 1986) and expanding into applications in text analysis. Recently, in part spurred by the resurgence of neural network methods, vector representations have enjoyed renewed attention, expanding beyond their previous scope with the appearance of a variety of new techniques for their generation and applications for their use.

The various approaches have been shown to have a variety of strengths and weaknesses and it is precisely in the context of this proliferation that our work is focused. Presented with diversity of techniques, other areas of machine learning have found excellent results with the use of ensemble methods (Dietterich, 2000), combining multiple techniques to capture the strengths of each. We examine whether similar gains are available here through the combination of multiple existing techniques for generating semantic vector representations.

Recent work has shown that relationships in these models (such as gender differences or pluralization) are often linear (Mikolov et al., 2013b). Drawing on this, we explore composition through linear combinations of these representational spaces. In particular, we explore combinations of a popular neural network method (Word2Vec) (Mikolov et al., 2013a) with Distributed Vector Representations in Sigma (DVRS) (Ustun et al., 2014), a method based on prior work in holographic representation (Jones and Mewhort, 2007). We demonstrate that various methods of composing these vectors can produce hybrid representations which perform significantly better than either method in isolation. This leap in performance is particularly pronounced when working with smaller datasets, opening up intriguing possibilities for domains which lack large corpora.

Proceedings of NAACL-HLT 2015, pages 95–101,
Denver, Colorado, May 31 – June 5, 2015.

2 Related Work

Most techniques for generating semantic vector spaces focus on the distributional properties of texts, building representations based on word co-occurrence patterns. Key factors differentiating these methods include the size of the context used in this process, and how the context is used.

Document-level representations consider co-occurrences at the document level, often making use of weighting factors such as *tf-idf*. Techniques such as Latent Semantic Analysis (LSA) (Deerwester et al., 1990) make use of this approach, building a word-document co-occurrence matrix for an entire corpus.

Other methods make use of a sliding window of words in a corpus, considering only words which occur within a certain distance of the target. The skip gram and continuous bag of words methods in Word2Vec are examples of this approach. Rather than a fixed width sampling space, other approaches make use of either sentence level (Levy and Goldberg, 2014a) (critical when using sentence-level parsing information) or paragraph windows, making use of the structure of the text itself to determine the window size.

Yet another class of methods makes use of randomly initialized word values with updates based on the local context. Techniques such as as BEAGLE (Jones and Mewhort, 2007) capture ordering information through the use of circular convolutions while DVRS makes use of piecewise vector multiplications over random projections.

As we make particular use of DVRS and Word2Vec, it is worth looking at the methods used by each in more detail.

2.1 DVRS

DVRS (Ustun et al., 2014) is a method for generating semantic vector representations based on ideas introduced in Jones and Mewhort (2007)'s BEAGLE system. Each word is represented by two vectors. The fixed environmental vector, $e(i)$, is randomly generated by drawing each element of the vector from a uniform distribution on $[-1, 1)$. The lexical vector $l(i)$ captures the word's distributional meaning and is updated over the course of training.

As the corpus is processed, the lexical vector for each encountered word is updated based on its paragraph context $c(k)$ and its sentence order context $o(k)$. The paragraph context is the sum of the environmental vectors in the paragraph, excluding the word being updated.

$$c(k) = \sum_{i=1}^{n} e(i), \text{where } i \neq k$$

The order context is based on the sentence and makes use of sequence vectors, random vectors which correspond the relative positions from the target word (by defaults ± 4). The word order information, $o(k)$, is then calculated as follows (where $*$ is the pointwise product operation):

$$o(k) = \sum_{j=-4}^{4} s(j) * e(k+j),$$

where $j \neq 0$ and $0 < (k+j) \leq n$.

DVRS updates via gradient descent where the gradient is based on the weighted sum of the order and context vectors.

$$l(k) = l(k) + w_c \hat{c(k)} + w_o \hat{o(k)}$$

In this update process, we see a precursor of the compositional techniques described below in that DVRS is already effectively composing two represenations (the context and order spaces) into a combined representation.

2.2 Word2Vec

The Word2Vec model learns word representations through a pair of architectures similar to standard feedforward neural net language models. The Continuous Bag-of-Words (CBOW) model effectively averages the vectors of all the words in a given context. The model is trained by predicting the current word based on the projected average of the surrounding context. The continuous Skip-gram model is similar, but instead of predicting the current word based on context, predicts the surrounding words based on the current. Words within a certain distance before and after the current word are predicted with the network optimized for these predictions.

3 Experiment 1: Methods of composing word representations

We extend the underlying concept of DVRS, the notion that multiple vector spaces can be linearly com-

bined to provide more effective distributional representations, by exploring the possibility of combining vector representations from completely separate generation techniques and the impact of different methods of composition. In particular, we explore the composition of Word2Vec and DVRS, effectively combining vectors produced by a neural network approach (particularly the predictive approach provided by the skip-gram model) with the larger context and count-based co-occurrence aggregation of the DVRS model. Across these components, we compare the compositional techniques of direct vector addition with vector concatenation.

We began by exploring alternative methods of combining vector representations. In particular, we looked at the differences between vector addition and concatenation. To get a better sense of the strengths and limits of these methods, we also compared against an oracle method where each of the individual models was tested and, if either was able to provide the correct answer, the oracle provided the answer from that model (exploring a hypothetically ideal simple combination of the two models).

For each of the following methods, we trained vectors against both the first 10^8 bytes of a Wikipedia dump from March 3, 2006 (enwik8) and the first 10^9 bytes of the same dump (enwik9)[1]. The data was initially pre-processed to convert all text to lower case, convert numbers to text, and eliminate links and other references[2].

For Word2Vec, we used the default settings (window size of 10, negative sampling (25), sampling (1e-4), and trained over 15 iterations) to create vectors of 300 and 1024 dimensions using both the skip gram and CBOW models. For DVRS, we created vectors of both 300 and 1024 dimensions, both using the default DVRS combination of order and context vectors and each of these components separately. These numbers were chosen based on previously published results where these values produced the best results for DVRS (1024 dimensions) and Word2Vec (300) respectively.

We tested combining these representational spaces through concatenation and addition. Addition required that the two spaces have the same rank and was accomplished using simple vector addition. The value for each word in the new space was the normalized sum of its two component spaces.

$$x_{new} = x_1 + x_2$$

We compared this against the direct concatenation of the two vectors for a given word. This allowed for the use of vectors with different underlying ranks.

$$x_{new} = x_1 \frown x_2$$

3.1 Results

There are a number of ways of evaluating the quality of vector representations. The space can be evaluated directly, for example by considering target words in terms of their evaluated similarity to their nearest neighbors in the space. Alternately, the space itself can be evaluated in terms of its usefulness as a feature for a known task.

We make use of the set of approximately 20,000 word analogy tests introduced by Mikolov et al (Mikolov et al., 2013a). Covering a mix of semantic and syntactic categories, each problem is based on a simple pattern of the form "A is to A* as B is to B*." Given A, A*, and B, the system is required to predict the correct B*. The test set includes 8,869 semantic test instances (with categories such as national capitals of the form "Athens is to Greece as Helsinki is to [Finland]") and 10,675 syntactic test instances (with categories such as superlatives such as "strong is to strongest as warm is to [warmest]"). We randomly split these tests, reserving 20% of the data for development and optimization and making use of the remaining 80% (15646 questions) for the test results below.

For experiment 1, the results are summarized in Table 1 (representations trained on the enwik8 dataset) and Table 2 (representations trained on the enwik9 dataset). For enwik8, among the individual-model directly-trained vectors, the Word2Vec skip gram model produced the best results with an overall accuracy of 30% on the test set compared to DVRS' best perfromance of 27%. However, while the skip gram model was always the top performer on the syntactic categories, DVRS turned in the best results on the semantic categories.

We then explored several methods of combining these vectors with the results summarized in Table 1. The addition of the 300-dimensional DVRS

[1]Obtained from `http://cs.fit.edu/~mmahoney/compression/textdata.html`.

[2]We used the script provided with the Wikipedia dump by Matt Mahoney.

vectors (overall performance 0.23) with the 300-dimensional skip gram vectors (overall performance 0.30) led to an overall improvement to 0.32. However, concatenating these two vectors improved the overall performance to 39%. Concatenating the best two performing vector sizes for each method (DVRS-1024, SG-300), yielded a slightly below peak overall accuracy of 0.38. Finally, we compared these results to an oracle method where, rather than composing the representations, we had each representation make its prediction and, if either was correct, used that as the oracle's response. This yielded an overall accuracy of 41%.

Next, we tested the methods against enwik9 as seen in Table 2. With additional training data, the performance of both methods jumped, with Word2Vec's skip gram model improving to an accuracy of 0.64 and DVRS improving to 0.43. While DVRS no longer showed the strong advantage on the semantic categories, nonetheless the concatenation of the two results showed a significant improvement to 0.67.

4 Experiment 2: Impacts of varying model parameters and input data

Following up on the initial experiment, we began to explore how much of the differences came from the structures of the algorithms versus the data available to the models. We varied the size of the context available to both Word2Vec and DVRS and the window size used by Word2Vec. In particular, we varied the parameters available to the best performing combinations from the previous experiment. Given the results of our prior experiment, we focused on concatenation rather than additive combinations.

We trained vectors against two versions of the enwik9 dump, one divided into paragraphs, the other into sentences. For Word2Vec, we varied the window size from 10 to 50, testing against both paragraph and sentence versions. Given the expanded window size, we chose to include the continuous bag of words (CBOW) model for Word2Vec. This is an alternative to the skip gram model which does not take into account word ordering. Given this, it seemed like this model might do better given the larger window size. DVRS structurally makes use of the full context available, so running it against paragraphs and sentences provided an analogous shift in effective window size.

4.1 Results

For this experiment, the results are summarized in Table 3. As seen in the previous experiment, the concatenated models generally did better than the component models. While window size made a slight difference for all the models, the differences in the algorithms seemed more important (with the caveat that more data is required for a definitive statement on this). In particular, the Word2Vec models and DVRS responded in opposite directions to having access to a larger window of data. DVRS performance improved slightly (from 0.407 to 0.409) while the Word2Vec models degraded slightly (for skip grams, from 0.640 to 0.594). This seems natural given that both Word2Vec models make use of sampling. Given that, a larger window simply distributes those samples over a larger and, likely, less meaningful space. Meanwhile, DVRS combines the complete context window to which it has access into a single factor, making larger windows more valuable to it (although at the cost of obscuring more local relations such as syntactic factors).

Nonetheless, as seen with DVRS where, in spite of inferior overall performance, it still proved effective as a concatenative element, we wanted to explore whether Word2Vec-trained vectors with larger windows might prove more effective in an ensemble. The results suggest that this may be the case. In particular, the concatenation of the two best performing Word2Vec models (the skip gram and CBOW models trained on sentences with a window size of 10) performed slightly worse (accuracy 0.655) than models which blended the best individual Word2Vec vectors (skip gram, sentence, window 10) with others (CBOW, paragraph, window 50) trained on a larger window (accuracy 0.664). However, the results here are sufficiently close that it will require more tests on a larger range of data before a general rule can be suggested.

Overall, the best performing concatenations were those where the differences in the two components was greatest. So, within Word2Vec-trained vectors, the skip gram with window 10 trained on sentences with the CBOW model with window 50 trained on paragraphs (0.664). As in the previous experi-

	Vector size	Overall	Semantic	Syntactic
DVRS	300	0.23	0.31	0.15
DVRS	1024	0.27	**0.40**	0.17
Skip gram (SG)	300	**0.30**	0.24	**0.34**
SG	1024	0.25	0.17	0.32
Add DVRS-300, SG-300	300	0.32	0.31	0.33
Concatenate DVRS-300, SG-300	600	**0.39**	**0.38**	0.39
Add DVRS-1024, SG-1024	2048	0.36	0.33	0.38
Concatenate DVRS-1024, SG-1024	2048	0.38	0.36	**0.40**
Concatenate DVRS-1024, SG-300	1324	0.38	**0.38**	0.39
Oracle DVRS-1024, SG-300	1024/300	0.41	0.44	0.39

Table 1: Performance on word analogy problems with vectors trained against the first 10^8 bytes of Wikipedia.

	Vector size	Overall	Semantic	Syntactic
DVRS	300	0.41	0.59	0.26
DVRS	1024	0.43	0.62	0.28
SG	300	**0.64**	**0.69**	**0.60**
SG	1024	0.57	0.60	0.55
Add 300-DVRS, 300-SG	300	0.64	0.72	0.58
Concatenate 300-DVRS, 300-SG	600	**0.67**	**0.74**	**0.60**
Add 1024-DVRS, 1024-SG	1024	0.60	0.66	0.55
Concatenate 1024-DVRS, 1024-SG	2048	0.61	0.68	0.55
Concatenate DVRS-1024, SG-300	1324	0.66	0.73	**0.60**
Oracle DVRS-1024, SG-300	1024/300	0.70	0.79	0.62

Table 2: Performance on word analogy problems with vectors trained against the first 10^9 bytes of Wikipedia.

ments, the best overall result (although marginally so) came from the concatenation of skip gram, sentence, window 10 with DVRS trained on paragraphs (0.671). Finally, even in cases where we were combining vectors from the same model but with different window sizes, we still found a small but consistent improvement in overall performance (for skip grams from 0.640 to 0.662, for CBOW from 0.635 to 0.660, and for DVRS from 0.409 to 0.426).

Care must be taken not to take these results as a claim that a specific combination recipe is correct or preferred. Instead, what our results show convincingly is that the combination of diverse representations can leverage strengths of individual representations, and that the effects of vector combination should be investigated in the context of specific tasks, which we leave as future work.

5 Discussion and Future Work

The question remains of how and why these compositions are working. While we do not claim a final answer to this question, we can point to several factors given these experiments. For the concatenated composition technique, we are left with the two vectors from the original spaces normalized into a single vector. For cosine similarity, the key factor is the dot product. For two vectors x_c and y_c which were formed by concatenating the corresponding vectors x_1 and x_2 from the original spaces, this yields:

$$x_c \cdot y_c = x_1 \cdot y_1 + x_2 \cdot y_2$$

As such, cosine similarity in the concatenated space is determined by a linear combination of the dot products of the component vectors. This provides an intuitive story for some of the behaviors

	Vector size	Overall	Semantic	Syntactic
skip gram (SG), paragraph (para), window 50	200	0.596	0.648	0.554
SG, sentence (sent), win 50	200	0.594	0.648	0.550
SG, sent, win 10	200	**0.640**	**0.682**	**0.607**
CBOW, para, win 50	200	0.603	0.672	0.547
CBOW, sent, win 50	200	0.599	0.661	0.548
CBOW, sent, win 10	200	0.635	0.673	0.604
DVRS, sent	300	0.407	0.590	0.259
DVRS, para	300	0.409	0.592	0.261
concat SG sent win 10, CBOW sent win 10	400	0.655	0.711	0.609
concat SG sent win 10, CBOW para win 50	400	0.664	0.735	0.607
concat SG sent win 10, SG para win 50	400	0.662	0.720	**0.614**
concat CBOW sent win 10, CBOW para win 50	400	0.660	0.727	0.605
concat DVRS para, SG sent win 10	500	**0.671**	**0.744**	0.612
concat DVRS sent, DVRS para	600	0.426	0.616	0.273
concat DVRS sent, CBOW para win 50	500	0.645	0.741	0.566
concat DVRS sent, SG sent win 10	500	0.666	0.739	0.606

Table 3: Variations on window size and data structure with vectors trained against the first 10^9 bytes of Wikipedia.

seen in that a close match with the correct answer in one space will tend to overcome drift in the other. However, a precise accounting of the variations in this behavior is one area where further work is required. In particular, exploring the impact of variations in the weights of this linear combination (easily done simply by weighting one of the vectors prior to concatenation) is an obvious first step. Additionally, it will be interesting to explore the combination of more than two vectors, effectively defining a new semantic space over those bases.

Generally, in the continuing discussion about the relative merits of count-based and prediction-based methods (Baroni et al., 2014), the present work suggests that there may not be a need to choose. By combining both methods through simple compositional functions, we show that it is possible to combine the benefits of both models in a single hybrid representation. Given the extensive work put into the development of distributed representations and the known variations in relative strengths and weaknesses, the benefit of these simple combination schemes is intriguing. We plan to explore the effects of vector combination in downstream tasks.

This work provides several key initial pieces. The first is an existence proof that, even with the most basic approaches and settings, it is possible to improve on the performance of individual models with only minimal increases in system complexity. Second, these experiments demonstrate that the local performance of a given representation is not necessarily a complete representation of its value as a component. In particular, inferior representations may still encode information which proves valuable to an ensemble. Finally, the observed improvements when vectors trained using the same method but with differing window sizes suggests that it may be possible to improve the performance of these algorithms (both Word2Vec models and DVRS) by making use of these observations, perhaps through the use of variable window sizes. In particular, we do not suggest that vector combination is the only way to achieve these improvements. There are multiple routes to incorporating this information within a single method.

One clear next step is to explore optimizing the weights of the components of the combinations in the context of particular tasks. Additionally, given recent discussions over alternative similarity measures (Levy and Goldberg, 2014b), it will be interesting to explore the generalization across spaces where non-linear composition may be re-

quired. More generally, the combination of representations may point towards tensor methods where multiple factors are preserved in the tensor structure itself.

Acknowledgments

We thank the anonymous reviewers for insightful suggestions. The effort described here has been partially sponsored by the U.S. Army. Any opinions, content or information presented does not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer.

Michael N Jones and Douglas JK Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, 114(1):1.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, Baltimore, Maryland, USA, June. Association for Computational Linguistics*.

James L McClelland, David E Rumelhart, PDP Research Group, et al. 1986. Parallel distributed processing. *Explorations in the microstructure of cognition*, 2.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Workshop Proceedings of the International Conference on Learning Representations*.

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

Volkan Ustun, PaulS. Rosenbloom, Kenji Sagae, and Abram Demski. 2014. Distributed vector representations of words in the sigma cognitive architecture. In Ben Goertzel, Laurent Orseau, and Javier Snaider, editors, *Artificial General Intelligence*, volume 8598 of *Lecture Notes in Computer Science*, pages 196–207. Springer International Publishing.