Improved Word Representations Via Summed Target and Context Embeddings

Nancy Fulda
Dept. of Computer Science
Brigham Young University
Provo, UT, USA
nfulda@cs.byu.edu

Nathaniel Robinson
Dept. of Computer Science
Brigham Young University
Provo, UT, USA
nrobinson@byu.edu

***Abstract*—Neural embedding models are often described as having an 'embedding layer', or a set of network activations that can be extracted from the model in order to obtain word or sentence representations. In this paper, we show via a modification of the well-known word2vec algorithm that relevant semantic information is contained throughout the entirety of the network, not just in the commonly-extracted hidden layer. This extra information can be extracted by summing embeddings from both the input and output weight matrices of a skip-gram model. Word embeddings generated via this method exhibit strong semantic structure, and are able to outperform traditionally extracted word2vec embeddings in a number of analogy tasks.**

I. Introduction

Word embeddings that encode accurate semantic meaning are in demand in many areas of Natural Language Processing (NLP, see section II). It is common practice to evaluate the semantic quality of an embedding by subjecting it to a variety of analogical reasoning tasks. These aim to evaluate whether the vector representations of similar words are cosine-close and how well semantic relationships are accurately encoded.

Many embedding algorithms learn two sets of embedding vectors. These sets are pulled from two different layers of the embedding network. Researchers often refer to one of these as the *embedding layer*, or the set of network activations that are extracted and used as the actual embedding vectors. This set of activations is also referred to as the *input weights* or the "target embeddings". The embeddings learned in the other layer of the network, called "output weights" or "context embeddings" are typically discarded. (We refer to this practice as "standard" throughout the paper.)

This approach is pragmatic in that it produces a concise, low-dimensional representation that is easy to conceptualize, but it may neglect useful information contained in the output weights. A similar problem has been observed, for example, in variational autoencoders; the decoder portion of the network takes on the burden of learning the target task, rendering the encoder's latent variables nearly useless [1]. Similarly, relevant semantic information is sometimes distributed throughout an embedding network rather than solely in the embedding layer.

In this paper we show that additional semantic information can be extracted from the word2vec skip-gram model introduced and popularized by [2]. We use the sum of the target and context embeddings as word vectors instead of the target embeddings only. The resulting embeddings, which we term word2vec-PLUS, have the same dimensionality as the target embeddings from the standard word2vec embedding layer, but they incorporate information from twice as many weight matrices. This information allows them to perform favorably on the SemEval 2013 and Google Analogy Test Set evaluation tasks and the SAT analogy task introduced by [3].

II. Background and Related Work

Vector space models, in which words or groups of words are represented as n-dimensional vectors, have been an active area of research since the 1960s [4]. Originally based on statistical models and co-occurrance counts, they have played a critical role in fields such as NLP and topic modeling.

A. Word and Sentence Embeddings

Recently, statistical models have been replaced by neural embeddings trained from large raw corpora. Mikolov et. al's [2] skip-gram model is among the most popular of these neural approaches. In this model, a matrix of input activations accepts a one-hot vector representing a target vocabulary word. This feeds into a single hidden layer, which, after multiplication by a matrix of output activations, feeds into an output layer representing the likelihood that each word in the vocabulary will appear within a context window of the input word (See Figure 1 (top)). Note that since matrix multiplication of the one-hot vector simply extracts the target word's corresponding row from the input weight matrix, the likelihood value for any pair of words is derived from the dot product between the target word's corresponding row in the input weight matrix and the context word's corresponding column in the output weight matrix.

Word vectors trained using the word2vec model have been shown to perform well at analogical reasoning tasks including the Google Analogy Test Set [5], affordance detection [6], and household item localization [7]. They have also proven useful in semantic analysis of language drift over time [8].

Other popular neural embedding models include the unsupervised log-bilinear model GloVe [9]; the FastText embedding algorithm, which uses subword information to accelerate training and account for out-of-vocabulary words [10]; and

BERT, a transformer-based model for learning contextualized word embeddings [11].

It is desirable for single-word embedding vectors to have strong semantic structure. Many of their uses, such as text categorization [12] and clinical NLP in medical applications [13], rely on the assumption that similar words will have vector representations that are cosine-close.

Neural embedding models have also been developed to encode multi-word inputs such as phrases, sentences, or documents. Some of the most well-known include InferSent [14], Google's universal sentence encoder [15], Sent2Vec [16], and skip-thought vectors [17]. Although these multi-word models are not the focus of our current work, they are susceptible to the same weaknesses as their single-word counterparts, and may benefit from the same solutions.

B. Previous Experiments Employing Context Embeddings

We are not the only researchers to investigate value of the information contained in context embeddings. Gabor et al. [18] use both target and context embeddings in a novel distance metric for word pair similarities. Other researchers have noted that words with proximate target vectors tend to be similar in different ways from word pairs in which the target vector of one is close to the context vector of the other (see section VI). Melamud et al. [19] leveraged this observation to improve automated lexical substitutions, and Nalisnick et al. [20] leveraged the same observation to improve document ranking with word embeddings. See also [12], a follow-up project by the same authors with more experimental results. Our work differs from the foregoing in that we use the sum of target and context embeddings to improve the semantic structure of the word embedding space itself, rather than applying the insight to the evaluation and interpretation of word vector proximity.

III. Word Representations via Summed Target and Context Embeddings

Our methodology for extracting word embeddings from a trained skip-gram model relies on the addition of vectors from each of the network's weight matrices. We call this algorithm word2vec-PLUS.

Traditionally, the input weights of Mikolov et al.'s skip-gram architecture are extracted to create word2vec embeddings. Thus the ith target vector is precisely ith row of the input weight matrix, where i is the vocabulary index of the target word [21]. Our key observation and contribution is that these input weights contain only part of the semantic information associated with the context-prediction task. The remainder of the information is located in the set of output weights, which connect the hidden layer to a set of output nodes that use Hierarchical SoftMax to generate a probability distribution over possible context words. Because of reliance on the Hierarchical Softmax on the output nodes and the association of multiple words with each hidden node on each forward pass (as opposed to only a single associated word from the input weights), this information is not as semantically well-structured as that contained in the input weights. Nevertheless it provides complimentary data that, when summed with weights from the input layer, is able to produce word embeddings with improved performance at semantic tasks.

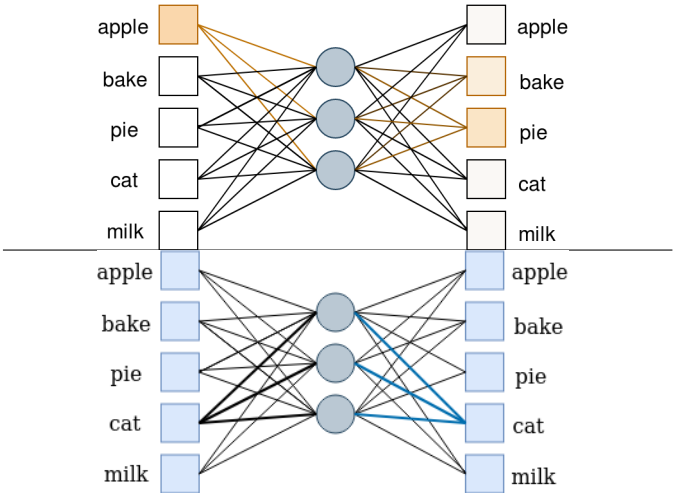

Fig. 1. *Top:* Forward pass of the word2vec skip-gram model. Colored squares indicate activation levels. In this example, the word 'apple' has been encoded as a one-hot vector. The weights associated with this word determine the values of the hidden layer, which in turn contribute to the probability values in the output layer. *Bottom:* Extraction of word2vec-PLUS embeddings for the word 'cat' from a pre-trained skip-gram model. Edges in the graph represent weights, with input weights bolded in black to the left and output weights bolded in green to the right.

A. Methodology

Our embedding extraction mechanism is shown in Figure 1 (bottom). The embeddings we use are from the sum of the input weights and the transpose of the output weights. A similar idea is employed by GloVe [9]. Formally, the vector representation $vec(i)$ of the ith vocabulary word can be described as:

$$vec(i) = W_{in}[i] + W_{out}^{T}[i] \tag{1}$$

Where W_{in} represents the input weights, W_{out} represents the output weights, and $W[n]$ denotes the nth row of the weight matrix W.

We also conducted tests on vectors produced by the output weights only, as well as concatenations rather than sums of the two sets of weights. The results we observed were underwhelming compared to those of summed embeddings, and we found the concatenation approach less desirable because it fails to preserve embedding dimension. These ideas are discussed further in section VI.

B. Training Corpora

We trained word2vec-PLUS embeddings on several corpora to evaluate the effectiveness of our algorithmic contribution across a variety of training sets. The corpora used are shown in Table I. Often the training sets we used were combinations of these corpora (via simple text concatenation).

Corpus	Size	Token count
Scraped articles	59.0 GB	9.6B
Wikipedia text	16.7 GB	2.8B
Toronto Book Corpus	4.6 GB	984M
Gutenberg classic books	1.2 GB	82M
Classic books (small)	20.3 MB	<1M

TABLE I

CORPORA USED TO TRAIN OUR EMBEDDING MODEL: A CORPUS OF SCRAPED WEB ARTICLES [22], A CLEANED WIKIPEDIA TEXT CORPUS [23], THE TORONTO BOOK CORPUS [24], A PRE-CONSTRUCTED CORPUS OF CLASSIC BOOKS FROM PROJECT GUTENBERG [25], AND A SMALLER CORPUS OF CLASSIC BOOKS FROM PROJECT GUTENBERG THAT WE COLLECTED IN HOUSE. THROUGHOUT THE PAPER WE REFER TO THESE CORPORA IN SHORTHAND AS "SCRAPED", "WIKIPEDIA", "BOOK", "GUTENBERG", AND "CLASSIC", RESPECTIVELY.

The corpus of scraped articles described in the caption of Table I was collected via an open-source method intended to clone the unreleased WebText dataset used to train OpenAI's GPT-2 [22] [26]. The Wikipedia corpus is the entirety of Wikipedia in 2004 with capital letters replaced. The Toronto Book Corpus is a collection of 11,038 books collected in a study by the University of Toronto [24]. The corpus entitled "Gutenberg classic books" is a set of 3,036 books from 142 different authors. See [25] for details about its collection methods.

IV. EXPERIMENTS

We tested word2vec-PLUS embeddings on three semantic evaluation tasks: The Google Analogy Test Set [27], SemEval 2013 [28], and the SAT Test Set [3]. Here we present our results in the context of performance of three well known single-word embedding algorithms: GloVe [9], FastText [10], and BERT [11].

A. Google Analogy Test Set

The Google Analogy Test Set [27] is commonly used to measure embedding quality. It was used to evaluate performance of both GloVe and FastText algorithms in their original publications [9] [10]. It contains 19,544 analogy questions in 14 categories. Eight of the categories contain questions about syntactic analogies, such as adjective-to-superlative (e.g. *fast:fastest :: small:?*). The remaining six have questions about semantic analogies, such as country-to-capital or antonym relationships.

In general, the questions in the Google Analogy Test Set are not cognitively complex. Any English-speaking human could answer most of the questions without difficulty. Notably, its questions are open-ended rather than multiple choice. In order to answer a question via word embeddings, we must search through all possible words in the embedding space and choose the best one.

To solve the questions in the Google Analogy Test Set, we apply the linear offset method. Given an analogy of the form *a:b::c:d*, we use the formula $\hat{d} = c + b - a$ and search the embedding space for the solution s that maximizes cosine similarity $sim_{cos}(\hat{d}, s)$ (the nearest vector neighbor to the calculated result).

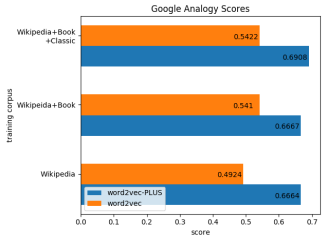

Fig. 2. Performance of word2vec-PLUS embeddings compared to standard word2vec extraction methods. Values displayed are the proportion of correctly answered test questions. On all three training corpora explored for this task, word2vec-PLUS embeddings outperform embeddings extracted using only the input weights of the skip-gram model. For this particular task training on larger corpora than those listed here resulted in prohibitively large vocabulary sizes that made a nearest neighbor search across the entire vector space less effective.

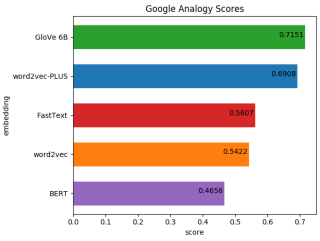

Fig. 3. Performance of different embedding algorithms on the Google Analogy Test Set. Word2vec-PLUS vectors were trained using a combined corpus containing data form Wikipedia, the Toronto Book Corpus, and Classic books. Throughout our study we used GloVe vectors trained on a 6B-token corpus and FastText trained on a 16B-token corpus, as these sizes were comparable to the size of corpora we used in word2vec-based training. Throughout our experiments we used zero-context BERT embeddings for analogy tasks.

Stem:		mason:stone
Choices:	(a)	teacher:chalk
	(b)	carpenter:wood
	(c)	soldier:gun
	(d)	photograph:camera
	(e)	book:word
Solution:	(b)	carpenter:wood

Fig. 4. Sample question from the SAT Dataset [29]. These questions are semantically more complex than the Google Analogy Test Set. Human performance on this task is generally around 55% [30].

Results are shown in Figures 2 and 3, and suggest that word2vec-PLUS embeddings encode significant semantic information. Not only do they outperform traditional word2vec embeddings across a variety of training corpora, but they are also able to exceed the performance of FastText vectors and nearly match the performance of 300-dimensional GloVe vectors. This is a significant result, noting that GloVe and FastText vectors were trained and tuned for this same analogy task. These results suggest that the usually discarded output weights in the skip-gram model contain valuable information.

B. SAT test set

The SAT Test Set [3] contains a list of 374 analogy questions used on the SAT. A sample SAT question is pictured in Figure 4. These questions are purely semantic and test an embedding space's ability to detect subtler relationships between less common words.

Note the questions are more cognitively difficult than the Google questions. The average human score on SAT analogy questions is close to 55 percent [30]. Also note that

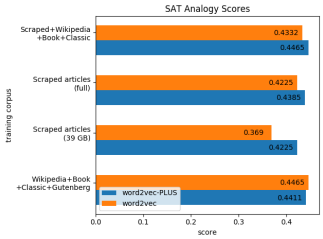

Fig. 5. Performance of word2vec-PLUS vectors compared with the standard word2vec embedding vectors on SAT. Note that word2vec-PLUS improves performance when embeddings are trained on larger albeit less informative text corpora such as the corpus of scraped articles, but it does not seem to help when the embeddings are trained on more informative albeit smaller texts.

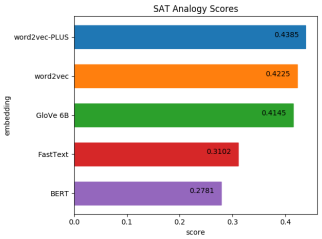

Fig. 6. Performance of different embedding algorithms on SAT analogy questions. The word2vec and word2vec-PLUS scores here were trained on the corpus of scraped articles since it resembles the Common Crawl data used to train GloVe [9] more closely than the other, more specialized corpora we explored. Note word2vec is particularly well-equipped for this task, and employing word2vec-PLUS improves on that score.

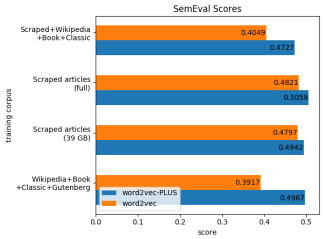

Fig. 7. Performance of word2vec-PLUS vectors compared with standard word2vec on SemEval task. The scores here are Pearson correlation r values. Higher value corresponds to closer correlation with human-provided responses. Note that word2vec-PLUS embeddings consistently outperform embeddings produced in the standard way.

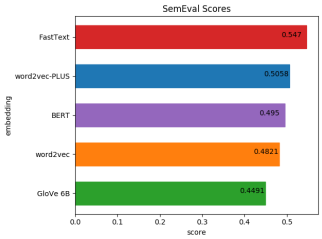

Fig. 8. Performance of different embedding algorithms on the SemEval sentence similarity questions. Scores for word2vec and word2vec-PLUS are taken from embeddings trained on the corpus of scraped articles as in Figure 6.

the questions are multiple-choice, so to answer them with embeddings we only need compare the vector representations of a few words. This makes the SAT test more difficult than the Google test in that the questions require deeper semantic knowledge but easier in that few options make the right choice more likely. SAT results for the different training corpora and embeddings schemes we observed are in Figures 5 and 6.

C. SemEval 2013

SemEval 2013 [28] is a collection of questions about sentence similarities. 1,118 sentence pairs were collected and given human-evaluated similarity scores via crowd-sourcing. To solve the SemEval task using word embeddings, we add the vectors of the words in each sentence and compare the cosine similarity of the two sentence-vectors for each question. The machine's score is the correlation between the embedding-produced cosine ratings and the human-produced ratings. This indicates the extent to which the embedding model is able to recognize sentence similarity in the same way that humans do. We note that multi-word embedding models would be naturally well-suited for this task. Our objective here is to analyze the effectiveness of our single-word-level embedding model at solving this complex semantic challenge. Accordingly, we restricted our experimental comparisons to other single-word embedding models. We assess the performance of word2vec-PLUS vectors in this task in Figures 7 and 8.

D. Hyperparameters

We performed a coarse search of hyperparameters including window size, embedding dimension, number of training epochs, and minimum word count for adoption into the vocabulary. We found that for our training corpora, a small window size of 2 resulted in better-performing embedding vectors for both standard word2vec and word2vec-PLUS embeddings. In the word2vec skip-gram algorithm the window size hyperparameter is actually used as an upper bound for the functional

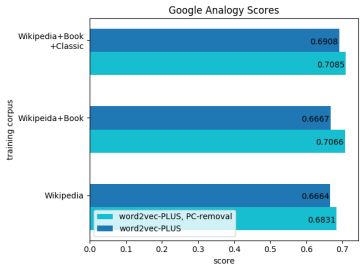

Fig. 9. Performance of word2vec-PLUS embeddings from different training corpora, with and without PC-removal. Note that PC-removal for word2vec-PLUS embeddings improves performance in the Google Analogy task.

window size, which is a random integer. It is conceivable that a lower bound made for more stability in the embedding space.

We also found in general that a lower minimum word count will yield better performance on SAT and SemEval benchmarks. In all of our reported results we used an embedding dimension of 300 (the same used for standard GloVe embeddings [9]) and trained for 3 epochs.

V. Principle Component Removal

Principle Component Removal (PC-removal) is an established technique used when creating sentence representations from single-word vectors [31]. We found that it is also useful in a single-word context, particularly when normalizing vectors comprised of summed weight slices (which may be distributed differently). We then re-assign each word vector v as $v \leftarrow uu^T v$, where u is the first singular vector of the embedding matrix. This results in a new set of word embeddings from which commonly shared properties have been removed. We found that removing the principle component from embedding vectors using this process consistently improved scores on the Google Analogy test. See Figure 9.

We also noticed that PC-removal resulted in a more dramatic improvement on the Google Analogy Test Set than on other evaluation tasks. This is perhaps because both the SAT test

and the SemEval test require deeper semantic knowledge that could be removed with the principle component. It is possible that PC-removal smooths noise that may detract from the bare-bones structural task presented by the simpler Google questions. It also appears that principle component removal is most helpful for embeddings that were trained on small corpora, suggesting that this might be an interesting bootstrapping method for low-resource domains.

Although PC-removal does not consistently improve SAT scores, it is sometimes beneficial. In fact the highest SAT score of any of the embeddings we trained was achieved by applying both word2vec-PLUS and principle component removal processes on the large combined corpus Scraped articles + Wikipedia + Toronto Book Corpus + Classic Books Collection. (The score was .4679.)

VI. Analysis of Test Results

We analyze some of the possible underlying reasons why word2vec-PLUS vectors may have more accurate word relationships in cosine distances and thus be better equipped for analogy tasks and other NLP applications.

The cosine similarity metric proportional to the dot product. Let a_{IN} be the target embedding vector (from the input weights) associated with word a, and let a_{OUT} be its context embedding vector (from the output weights). Then finding the similarity between words a and b via cosine distance varies, depending on whether our chosen embedding for a word is (1) the target embedding only, (2) the context embedding only, (3) the sum of the two, or (4) the concatenation of the two. The different similarity metrics for these schema are summarized below.

$$sim(a,b) \propto a_{IN}^T b_{IN} \tag{2}$$

$$sim(a,b) \propto a_{OUT}^T b_{OUT} \tag{3}$$

$$sim(a,b) \propto (a_{IN}^T + a_{OUT}^T)(b_{IN} + b_{OUT}) = a_{IN}^T b_{IN} + a_{IN}^T b_{OUT} + a_{OUT}^T b_{IN} + a_{OUT}^T b_{OUT} \tag{4}$$

$$sim(a,b) \propto [a_{IN}^T, a_{OUT}^T] \begin{bmatrix} b_{IN} \\ b_{OUT} \end{bmatrix} = a_{IN}^T b_{IN} + a_{OUT}^T b_{OUT} \tag{5}$$

As discussed briefly in Section 2, Nalisnick et al. [20] found that the term $a_{IN}^T b_{IN}$ is a measure of *typical* similarity between words a and b. This means if similarity is high, a and b are the same type of word and occur in similar contexts. They also found that the terms $a_{IN}^T b_{OUT}$ and $a_{OUT}^T b_{IN}$ are measures of *topical* similarity between words a and b. This means that if similarity is high, a and b are words of the same topic and are likely to occur in contexts together. From Nalisnick et al's work, examples of words *typically* similar to the word *yale* are *harvard*, *nyu*, and *cornell*, whereas words *topically* similar to *yale* are *faculty*, *alumni*, and *orientation*.

This observation by Nalisnick et al. is mathematically consistent with the way the skip-gram model is trained. The network's objective is to maximize the inner product $a_{IN}^T b_{OUT}$ when words a and b occur in the same context. (See section II.) This ensures that the value will be high for *topically* similar words. It also ensures theoretically that both $a_{IN}^T b_{IN}$ and $a_{OUT}^T b_{OUT}$ should be high for *typically* similar words; vectors (both target and context) for words that occur in similar contexts will become cosine-similar to the same set of vectors and thereby become cosine-similar to each other.

Note that the terms indicating *topical* similarity are present only when we use sum embeddings (like word2vec-PLUS). This may be why word2vec-PLUS embeddings tend to have an advantage in analogy tasks. Many analogy questions require an ability to discern both *typical* similarity and *topical* similarity. Consider the analogy *mason:stone :: carpenter:wood* from Figure 4. The words *mason* and *carpenter* (as well as *wood* and *stone*) are *typically* similar. The words *mason* and *stone* (as well as *carpenter* and *wood*) are *topically* similar. In order for an embedding space to predict that *wood* completes the analogy, it must have a vector representation for *wood* that is close to $v_{stone} + v_{carpenter} - v_{mason}$. More formally, the value $v_{wood}^T(v_{stone} + v_{carpenter} - v_{mason}) = v_{wood}^T v_{stone} + v_{wood}^T v_{carpenter} - v_{wood}^T v_{mason}$ must be high. Thus we see the embedding space must be equipped to know that *wood* is similar to both *carpenter* and *stone*.

An average of 54 different SAT tests for both summed vectors and concatenated vectors are shown in Table II.

sum	concatenate
.363	.349

TABLE II
AVERAGE SAT ACCURACY SCORES FOR BOTH SUMMED AND CONCATENATED VECTORS, TRAINED ON 7 SMALL CORPORA WITH VARYING HYPERPARAMETERS. BECAUSE CONCATENATED EMBEDDINGS PERFORMED WORSE IN THESE INITIAL EXPERIMENTS, AND BECAUSE OF THEIR UNDESIRABLY HIGH EMBEDDING DIMENSION, WE FOCUSED MORE ON SUMMED EMBEDDINGS IN THIS WORK.

VII. Conclusions and Future Work

In this paper, we have presented a new methodology for extracting semantic information from a trained skip-gram model. Rather than extracting the input weights only, we sum them with the model's output weights corresponding to the same vocabulary words. This process improves performance across a variety of semantic analogy tasks.

A key aspect of this work is the empirical demonstration that additional extricable information exists within the weights and activation layers of the entire network, not just within the commonly-extracted embedding layer. This research may be expanded in a number of ways. The algorithmic adjustment we applied to the skip-gram word2vec model may be applied to other embedding algorithms such as FastText, GloVe, or other representational neural networks. Further experimentation with different ways to combine target and context embeddings to produce word vectors may yield a more optimal result than those presented here. Given the significant computational resources required to train large language models, future work in this area may focus on developing improved extraction

methods for more complex neural architectures, including transformers and other multi-word embedding models.

Evaluation of the summed embeddings presented here in multiple NLP tasks may yield further insight into their uses. Like other researchers ([32], [33]) who have suggested alternative quality metrics for word embeddings than performance on analogy tests, we do not claim that embeddings that are better at analogy tasks are necessarily better in all NLP applications. Much like Batchkarov et al. [33], we see analogy tasks as a useful tool to determine whether semantic word relationships are accurately encoded into an embedding and particularly whether cosine word relationships can be trusted.

REFERENCES

[1] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *International Conference on Learning Representations*, 2017.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Peter D Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

[4] Sheldon Klein, Stephen L Lieman, and Gary Lindstrom. Diseminer: a distributional-semantics inference maker. 1966.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, pages 3111–3119. Curran Associates, Inc., 2013.

[6] Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1039–1045, 2017.

[7] Nancy Fulda, Nathan Tibbetts, Zachary Brown, and David Wingate. Harvesting common-sense navigational knowledge for robotics from uncurated text corpora. In *Proceedings of the First Conference on Robot Learning*, 2017.

[8] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Association for Computational Linguistics (ACL)*, 2016.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014.

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

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*, 2016.

[13] A. Bagheri, A. Sammani, P.G.M. van der Heijden, and Oberski D.L. Asselbergs, F.W. Automatic ICD-10 classification of diseases from Dutch discharge letters. In *BIOINFORMATICS 2020 - 11th International Conference on Bioinformatics Models, Methods and Algorithms, Proceedings; Part of 13th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2020*, pages 281–289. Association for Computational Linguistics, 2020.

[14] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[15] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[16] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *6th International Conference on Learning Representations*, 2018.

[17] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *Advances in neural information processing systems*, pages 3294–3302, 2015.

[18] Kata Gábor, Haïfa Zargayouna, Isabelle Tellier, Davide Buscaldi, and Thierry Charnois. Exploring vector spaces for semantic relations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1814–1823, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[19] Oren Melamud, Omer Levy, and Ido Dagan. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado, June 2015. Association for Computational Linguistics.

[20] Eric Nalisnick, Mitra Bhaskar, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *WWW '16 Companion: Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84, 2016.

[21] Xin Rong. Word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[22] Joshua C. Peterson. OpenWebText. https://github.com/jcpeterson/openwebtext, 2019.

[23] Wikipedia contributors. Plagiarism — Wikipedia, the free encyclopedia, 2004. [Online; accessed 22-July-2004].

[24] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[25] Shibamouli Lahiri. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[26] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[27] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. Association for Computational Linguistics, 2013.

[28] Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Alan Ritter, Sara Rosenthal, and Veselin Stoyanov. Sentiment analysis in twitter. http://www.cs.york.ac.uk/semeval-2013/task2/, 2013.

[29] Peter D. Turney. Analogy perception applied to seven tests of word comprehension. *J. Exp. Theor. Artif. Intell.*, 23:343–362, 2011.

[30] T. Veale. *Exploding The Creativity Myth: The Computational Foundations of Linguistic Creativity*. Bloomsbury Publishing, 2012.

[31] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*, 2017.

[32] Anna Gladkova and Aleksandr Drozd. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42, Berlin, Germany, August 2016. Association for Computational Linguistics.

[33] Miroslav Batchkarov, Thomas Kober, Jeremy Reffin, Julie Weeds, and David Weir. A critique of word similarity as a method for evaluating distributional semantic models. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 7–12. Association for Computational Linguistics, August 2016. 1st Workshop on Evaluating Vector Space Representations for NLP, RepEval 2016 ; Conference date: 12-08-2016 Through 12-08-2016.