Bringing Order to Neural Word Embeddings with Embeddings Augmented by Random Permutations (EARP)

Trevor Cohen
Biomedical and Health Informatics
University of Washington, Seattle
cohenta@uw.edu

Dominic Widdows
Grab, Inc.
Seattle, WA
dominic.widdows@grab.com

Abstract

Word order is clearly a vital part of human language, but it has been used comparatively lightly in distributional vector models. This paper presents a new method for incorporating word order information into word vector embedding models by combining the benefits of permutation-based order encoding with the more recent method of skip-gram with negative sampling. The new method introduced here is called Embeddings Augmented by Random Permutations (EARP). It operates by applying permutations to the coordinates of context vector representations during the process of training. Results show an 8% improvement in accuracy on the challenging Bigger Analogy Test Set, and smaller but consistent improvements on other analogy reference sets. These findings demonstrate the importance of order-based information in analogical retrieval tasks, and the utility of random permutations as a means to augment neural embeddings.

1 Introduction

The recognition of the utility of corpus-derived distributed representations of words for a broad range of Natural Language Processing (NLP) tasks (Collobert et al., 2011) has led to a resurgence of interest in methods of distributional semantics. In particular, the neural-probabilistic word representations produced by the Skip-gram and Continuous Bag-of-Words (CBOW) architectures (Mikolov et al., 2013a) implemented in the `word2vec` and `fastText` software packages have been extensively evaluated in recent years.

As was the case with preceding distributional models (see for example (Schütze, 1993; Lund and Burgess, 1996; Schütze, 1998; Karlgren and Sahlgren, 2001)), these architectures generate vector representations of words — or word embeddings — such that words with similar proximal neighboring terms within a corpus of text will have similar vector representations. As the relative position of these neighboring terms is generally not considered, distributional models of this nature are often (and sometimes derisively) referred to as *bag-of-words* models. While methods of encoding word order into neural-probabilistic representations have been evaluated, these methods generally require learning additional parameters, either for each position in the sliding window (Mnih and Kavukcuoglu, 2013), or for each context word-by-position pair (Trask et al., 2015; Ling et al., 2015).

In this paper we evaluate an alternative method of encoding the position of words within a sliding window into neural word embeddings using Embeddings Augmented by Random Permutations (EARP). EARP leverages random permutations of context vector representations (Sahlgren et al., 2008), a technique that has not been applied during the training of neural word embeddings previously. Unlike prior approaches to encoding position into neural word embeddings, it imposes no computational and negligible space requirements.

Results show that the word order information encoded through EARP leads to a nearly 8% improvement (from 29.17% to 37.07%) in the accuracy of analogy predictions in the Bigger Analogy Test Set of Gladkova et al (2016), with the improvement being largest (over 20%) in the category of analogies that exhibit derivational morphology (a case where the use of subword information also improves accuracy for all representations with and without order information). Smaller improvements in performance are evident on other analogy sets, and in downstream sequence labeling tasks. This makes EARP a strong contender for enriching word embeddings with order-based information, leading to greater accuracy on more challenging semantic processing tasks.

2 Background

2.1 Distributed representations of words

Methods of distributional semantics learn representations of words from the contexts they occur in across large text corpora, such that words that occur in similar contexts will have similar representations (Turney and Pantel, 2010). Geometrically-motivated approaches to this problem often involve decomposition of a term-by-context matrix (Schütze, 1993; Landauer and Dumais, 1997; Pennington et al., 2014), resulting in word vectors of considerably lower dimensionality than the total number of contexts. Alternatively, reduced-dimensional representations can be generated online while processing individual units of text, without the need to represent a large term-by-context matrix explicitly. A seminal example of the latter approach is the Random Indexing (RI) method (Kanerva et al., 2000), which generates distributed representations of words by superposing randomly generated context vector representations.

Neural-probabilistic methods have shown utility as a means to generate semantic vector representations of words (Bengio et al., 2003). In particular, the Skip-gram and CBOW neural network architectures (Mikolov et al., 2013a) implemented within the `word2vec` and `fastText` software packages provide scalable approaches to online training of word vector representations that have been shown to perform well across a number of tasks (Mikolov et al., 2013a; Levy et al., 2015; Mikolov et al., 2017).

While the definition of what constitutes a context varies across models, a popular alternative is to use words in a sliding window centered on a focus term for this purpose. Consequently where decomposition is involved the matrix in question would be a term-by-term matrix. With online methods, each term has two vector representations — a *semantic vector* and a *context vector*, corresponding to the input and output weights for each word in neural-probabilistic approaches.

2.2 Order-based distributional models

In most word vector embedding models based on sliding windows, the relative position of words within this sliding window is ignored, but there have been prior efforts to encode this information. Before the popularization of neural word embeddings, a number of researchers developed and evaluated methods to encode word position into distributed vector representations. The BEAGLE model (Jones et al., 2006) uses circular convolution — as described by Plate (1995) — as a *binding operator* to compose representations of n-grams from randomly instantiated context vector representations of individual terms. These composite representations are then added to the semantic vector representation for the central term in a sliding window. The cosine similarity between the resulting vectors is predictive of human performance in semantic priming experiments.

A limitation of this approach is that it involves a large number of operations per sliding window. For example, Jones and his colleagues (2006) employ eight superposition and nine convolution operations to represent a single sliding window of three terms (excluding the focus term). Sahlgren, Holst and Kanerva (2008) report a computationally simpler method of encoding word order in the context of RI. Like BEAGLE, this approach involves adding randomly instantiated context vectors for adjacent terms to the semantic vector of the central term in a sliding window. However, RI uses sparse context vectors, consisting of mostly zeroes with a small number non-zero components initialized at random. These vectors are assigned permutations indicating their position within a sliding window. For example, with $\prod_p$ representing the permutation assigned to a given position p and words to the right of the equation representing their context vectors, the sliding window “[fast is **fine** but accuracy] is everything” would result in the following update to $S(\text{fine})$, the semantic vector for the term “fine”:

$$\begin{aligned} S(\text{fine}) \quad += \quad & \prod_{-2} \overrightarrow{\text{fast}} + \prod_{-1} \overrightarrow{\text{is}} \\ & + \prod_{+1} \overrightarrow{\text{but}} + \prod_{+2} \overrightarrow{\text{accuracy}} \end{aligned}$$

Amongst the encoding schemes evaluated using this approach, using a pair of permutations to distinguish between words preceding the focus term and those following it resulted in the best performance on synonym test evaluations (Sahlgren et al., 2008). Furthermore, this approach was shown to outperform BEAGLE on a number of evaluations on account of its ability to scale up to larger amounts of input data (Recchia et al., 2010).

2.3 Encoding order into neural embeddings

Some recent work has evaluated the utility of encoding word order into neural word embeddings.

A straightforward way to accomplish this involves maintaining a separate set of parameters for each context word position, so that for a vocabulary of size v and p context window positions the number of output weights in the model is $v \times p \times d$ for embeddings of dimensionality d. This approach was applied by Ling and his colleagues (2015) to distinguish between terms occurring before and after the central term in a sliding window, with improvements in performance in downstream part-of-speech tagging and dependency parsing tasks.

Trask and his colleagues (2015) develop this idea further in a model they call a Partitioned Embedding Neural Network (PENN). In a PENN, both the input weights (word embeddings) and output weights (context embeddings) of the network have separate position-specific instantiations, so the total number of model parameters is $2v \times p \times d$. In addition to evaluating binary (before/after) context positions, the utility of training separate weights for each position within a sliding window was evaluated. Incorporating order in this way resulted in improvements in accuracy over `word2vec`'s CBOW implementation on proportional analogy problems, which were considerably enhanced by the incorporation of character-level embeddings.

A more space-efficient approach to encoding word order involves learning a vector V for each position p in the sliding window. These vectors are applied to rescale context vector representations using pointwise multiplication while constructing a weighted average of the context vectors for each term in the window (Mnih and Kavukcuoglu, 2013; Mikolov et al., 2017). Results reported using this approach are variable, with some authors reporting worse performance with position-dependent weights on a sentence completion task (Mnih and Kavukcuoglu, 2013), and others reporting improved performance on an analogy completion task (Mikolov et al., 2017).

3 Methods

3.1 Skipgram-with-negative-sampling

In the current work, we extend the skipgram-with-negative-sampling (SGNS) algorithm to encode positional information without additional computation. The Skip-gram model (Mikolov et al., 2013a) predicts $p(c|w)$: the probability of observing a context word, c, given an observed word, w. This can be accomplished by moving a sliding window through a large text corpus, such that the observed word is at the center of the window, and the context words surround it. The architecture itself includes two sets of parameters for each unique word: The input weights of the network ($\overrightarrow{w}$), which represent observed words and are usually retained as semantic vectors after training, and the output weights of the network which represent context words ($\overrightarrow{c}$) and are usually discarded. The probability of observing a word in context is calculated by appying the sigmoid function to the scalar product between the input weights for the observed word, and the output weights for the context word, such that $p(c|w) = \sigma(\overrightarrow{w}.\overrightarrow{c})$. As maximizing this probability using the softmax over all possible context words would be computationally inconvenient, SGNS instead draws a small number of *negative samples* ($\neg c$) from the vocabulary to serve as counterexamples to each observed context term. With D representing observed term/context pairs, and D' representing randomly constructed counterexamples the SGNS optimization objective is as follows (Goldberg and Levy, 2014):

$$\sum_{(w,c)\in D} \log \sigma(\overrightarrow{w}.\overrightarrow{c}) + \sum_{(w,\neg c)\in D'} \log \sigma(-\overrightarrow{w}.\overrightarrow{\neg c})$$

3.2 Embeddings Augmented by Random Permutations (EARP)

Sliding window variants of RI are similar in some respects to the SGNS algorithm, in that training occurs through an online update step in which a vector representation for each context term is added to the semantic vector representation for a focus term (with SGNS this constitutes an update of the input weight vector for the focus term, weighted by the gradient and learning rate (for a derivation see (Rong, 2014)). Unlike SGNS, however, context vectors in RI are immutable and sparse. With SGNS, dense context vectors are altered during the training process, providing an enhanced capacity for inferred similarity. Nonetheless, the technique of permuting context vectors to indicate the position of a context term within a sliding window is readily adaptable to SGNS.

Our approach to encoding the position of context words involves assigning a randomly-generated permutation to each position within a sliding window. For example, with a sliding window of window radius 2 (considering two positions to the left and the right of a focus term), we assign a random permutation, $\prod_p$, to each element

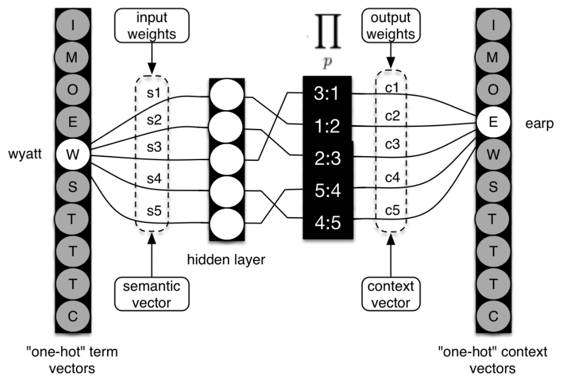

Figure 1: The Architecture of EARP

of the set of positions $\{-2; -1; +1; +2\}$. With $\prod_p$ indicating the application of a context-specific random permutation for position p, the optimization objective becomes:

$$\sum_{(w,c:p)\in D} \log\sigma(\vec{w}.\prod_p(\vec{c})) + \sum_{(w,\neg c:p)\in D'} \log\sigma(-\vec{w}.\prod_p(\vec{\neg c}))$$

For example, upon observing the term "wyatt" in the context of the term "earp", the model would attempt to maximize $p(c|w) = \sigma(\overrightarrow{\text{wyatt}}.\prod_{+1}(\overrightarrow{\text{earp}}))$, with $\overrightarrow{\text{wyatt}}$ and $\overrightarrow{\text{earp}}$ as semantic and context vectors respectively. The underlying architecture is illustrated in Figure 1, using a simplified model generating 5-dimensional vector representations for a small vocabulary of 10 terms. The permutation $\prod_p$ can be implemented by "rewiring" the components of the input and output weights, without explicitly generating a permuted copy of the context vector concerned. In this way, $p(c|w)$ is estimated and maximized in place without imposing additional computational or space requirements, beyond those required to store the permutations. This is accomplished by changing the index values used to access components of $\overrightarrow{\text{earp}}$ when the scalar product is calculated, and when weights are updated. The inverse permutation (or reverse rewiring - connecting components 1:3 rather than 3:1) is applied to $\overrightarrow{\text{wyatt}}$ when updating $\overrightarrow{\text{earp}}$, and this procedure is used with both observed context terms and negative samples.

Within this general framework, we evaluate four word order encoding schemes, implemented by adapting the open source `Semantic Vectors`[1] package for distributional semantics research:

[1]https://github.com/semanticvectors/semanticvectors

3.2.1 Directional ($EARP_{dir}$)

Directional encoding draws a distinction between terms that occur before or after the focus term in a sliding window. As such, only two permutations (and their inverse permutations) are employed: $\prod_{-1}$ for preceding terms, and $\prod_{+1}$ for all subseqent terms. Directional encoding with permutations has been shown to improve performance in synonym tests evaluations when applied in the context of RI (Sahlgren et al., 2008).

3.2.2 Positional ($EARP_{pos}$)

With positional encoding, a permutation is used to encode each space in the sliding window. As a randomly permuted vector is highly likely to be orthogonal or close-to-orthogonal to the vector from which it originated (Kanerva, 2009), this is likely to result in orthogonal encodings for the same context word in different positions. With RI, positional encoding degraded synonym test performance, but permitted a novel form of distributional query, in which permutation is used to retrieve words that occur in particular positions in relation to one another (Sahlgren et al., 2008). $EARP_{pos}$ facilitates queries of this form also: the nearest neighboring context vector to the permuted semantic vector $\prod_{-1}^{\text{INV}}(\overrightarrow{\text{earp}})$ is $\overrightarrow{\text{wyatt}}$ in both static-window subword-agnostic $EARP_{pos}$ spaces used in the experiments that follow.

3.2.3 Proximity-based ($EARP_{prox}$)

With proximity-based encoding, the positional encoding of a particular context term occurring in the first position in a sliding window will be somewhat similar to the encoding when this term occurs in the second position, and less similar (but still not orthogonal) to the encoding when it occurs in the third. This is accomplished by randomly generating an *index permutation* $\prod_{+1}$, randomly reassigning a half of its permutations to generate $\prod_{+2}$, and repeating this process iteratively until a permutation for every position in the window is obtained (for the current experiments, we assigned two index permutations $\prod_{+1}$ and $\prod_{-1}$, proceeding bidirectionally). As a low-dimensional example, if $\prod_{+1}$ were {4:1, 1:2, 2:3, 3:4}, $\prod_{+2}$ might be {4:3, 1:2, 2:1, 3:4}. The net result is that the similarity between the position-specific representations of a given context vector reflects the proximity between the positions concerned. While this method is reminiscent of interpolation between randomly generated vectors or matrices to encode character position within words (Cohen et al., 2013) and pixel position within images (Gallant and Culliton, 2016) respectively, the iterative application of permutations for this purpose is a novel approach to such positional binding.

3.3 Subword embeddings

The use of character n-grams as components of distributional semantic models was introduced by Schütze (1993), and has been shown to improve performance of neural-probabilistic models on analogical retrieval tasks (Bojanowski et al., 2017; Mikolov et al., 2017). It is intuitive that this should be the case as standard analogy evaluation reference sets include many analogical questions that require mapping from a morphological derivative of one word (e.g. fast:faster) to the same morphological derivative of another (e.g. high:higher).

Consequently, we also generated n-gram based variants of each of our models, by adapting the approach described in (Bojanowski et al., 2017) to our SGNS configuration. Specifically, we decomposed each word into character n-grams, after introducing characters indicating the start (<) and end (>) of a word. N-grams of size betweeen 3 and 6 characters (inclusive) were encoded, and in order to place an upper bound on memory requirements a hash function was used to map each observed n-gram to one of at the most two million vectors without constraints on collisions. The input vector V_i for a word with n included n-grams (including the word itself) was then generated as[2]

$$V_i = \frac{1}{n}\sum_{i=1}^{n} \overrightarrow{\mathrm{ngram_i}}$$

During training, we used V_i as the input vector for EARP and SGNS, with updates propagating back to component word and n-gram vectors in proportion to their contribution to V_i.

3.4 Training data

All models were trained on the first January 2018 release of the English Wikipedia[3], to which we applied the pre-processing script distributed with the `fastText` package[4], resulting in a corpus of approximately 8.8 billion words.

[2]Words and n-grams are weighted equally, following (Bojanowski et al., 2017) and the `fastText` implementation

[3]`https://dumps.wikimedia.org/enwiki`

[4]`https://github.com/facebookresearch/fastText`

3.5 Training procedure

All models used 500-dimensional vectors, and were trained for a single iteration across the corpus with five negative samples per context term. For each of the four models ($SGNS$, $EARP_{dir}$, $EARP_{pos}$, $EARP_{prox}$), embeddings were generated with sliding window radii r of 2 and 5 (in each direction), with and without subword embeddings. For all experiments, we excluded numbers, and terms occurring less than 150 times in the corpus. SGNS has a number of hyperparameters that are known to influence performance (Levy et al., 2015). We did not engage in tuning of these hyperparameters to improve performance of our models, but rather were guided by prior research in selecting hyper-parameter settings that are known to perform well with the SGNS baseline model.

Specifically, we used a subsampling threshold t of 10^{-5}. In some experiments (*EARP* and *SGNS*), we used dynamic sliding windows with uniform probability of a sliding window radius between one and r, in an effort to align our baseline model closely with other SGNS implementations. Stochastic reduction of the sliding window radius will result in distal words being ignored at times. With a dynamic sliding window, subsampled words are replaced by the next word in sequence. This increases the data available for training, but will result in relative position being distorted at times. Consequently, we also generated spaces with static fixed-width sliding windows (*EARPx*) for position-aware models.

After finding that `fastText` trained on Wikipedia performed better on analogy tests than prior results obtained with `word2vec` (Levy et al., 2015), we adopted a number of its hyperparameter settings. We set the probability of negative sampling for each term to $f^{\frac{1}{2}}$, where f is the number of term occurrences divided by the total token count. In addition we used an initial learning rate of .05, and subsampled terms with a probability of $1 - (\sqrt{\frac{t}{f}} + \frac{t}{f})$[5].

[5]While using this formula reliably improves performance on some of the analogy sets, it differs from both the formula described in Mikolov et al. (2013b), and the formula implemented in the canonical `word2vec` implementation of SGNS - see Levy et al. (2015) for details. It is also difficult to justify on theoretical grounds, as it returns values less than zero for some words that meet the subsampling threshold. Nevertheless, retaining it throughout our experiments seemed more principled than altering the `fastText` baseline in a manner that impaired its performance.

3.6 Evaluation

To evaluate the nature and utility of the additional information encoded by permutation-based variants, we utilized a set of analogical retrieval reference sets, including the MSR set (Mikolov et al., 2013c) consisting of 8,000 proportional analogy questions that are morphological in nature (e.g. *young:younger:quick:?*) and the Google analogy set (Mikolov et al., 2013a) which includes 8,869 semantic (and predominantly geographic, e.g. *brussels:belgium:dublin:?*) and 10,675 morphologically-oriented "syntactic" questions. We also included the Bigger Analogy Test Set (BATS) set (Gladkova et al., 2016), a more challenging set of 99,200 proportional analogy questions balanced across 40 linguistic types in four categories: Inflections (e.g. plurals, infinitives), Derivation (e.g. verb+er), Lexicography (e.g. hypernyms, synonyms) and Encylopedia (e.g. country:capital, male:female). We obtained these sets from the distribution described in Finley et al (2017)[6], in which only the first correct answer to questions with multiple correct answers in BATS is retained, and used a parallelized implementation of the widely used vector offset method, in which for a given proportional analogy *a:b:c:d*, all word vectors in the space are rank-ordered in accordance with their cosine similarity to the vector $\overrightarrow{c+b-a}$. We report average accuracy, where a result is considered accurate if d is the top-ranked result aside from a, b and c.[7]

To evaluate the effects of encoding word order on the relative distance between terms, we used a series of widely used reference sets that mediate comparison between human and machine estimates of pairwise similarity and relatedness between term pairs. Specifically, we used Wordsim-353 (Finkelstein et al., 2001), split into subsets emphasizing similarity and relatedness (Agirre et al., 2009); MEN (Bruni et al., 2014) and Simlex-999 (Hill et al., 2015). For each of these sets, we estimated the Spearman correlation of the cosine similarity between vector representations of the words in a given pair, with the human ratings (averaged across raters) of similarity and/or relatedness provided in the reference standards.

Only those examples in which all relevant terms were represented in our vector spaces were considered. Consequently, our analogy test sets consisted of 6136; 19,420 and 88,108 examples for the MSR[8], Google and BATS sets respectively. With pairwise similarity, we retained 998; 335 and all 3,000 of the Simlex, Wordsim and MEN examples respectively. These numbers were identical across models, including `fastText` baselines.

In addition we evaluated the effects of incorporating word order with EARP on three standard sequence labeling tasks: part-of-speech tagging of the Wall Street Journal sections of the Penn Treebank (PTB) and the CoNLL'00 sentence chunking (Tjong Kim Sang and Buchholz, 2000) and CoNLL'03 named entity recognition (Tjong Kim Sang and De Meulder, 2003) shared tasks. As was the case with the pairwise similarity and relatedness evaluations, we conducted these evaluations using the `repEval2016` package [9] after converting all vectors to the `word2vec` binary format. This package provides implementations of the neural NLP architecture developed by Collobert and his colleagues (2011), which uses vectors for words within a five-word window as input, a single hidden layer of 300 units and an output Softmax layer. The implementation provided in `repEval2016` deviates by design from the original implementation by fixing word vectors during training in order to emphasize difference between models for the purpose of comparative evaluation, which tends to reduce performance (for further details, see (Chiu et al., 2016)). As spaces constructed with narrower sliding windows generally perform better on these tasks (Chiu et al., 2016), we conducted these experiments with models of window radius 2 only. To facilitate fair comparison, we added random vectors representing tokens available in the `fastText`-derived spaces only to all spaces, replacing the original vectors where these existed. This was important in this evaluation as only `fastText` retains vector representation for punctuation marks (these are eliminated by the `Semantic Vectors` tokenization procedure), resulting in a relatively large number of out-of-vocabulary terms and predictably reduced

[6] `https://github.com/gpfinley/analogies`

[7] For a finer-grained estimate of performance — which we considered particularly important in cases in BATS in which only the first of a set of correct answers was retained — we also calculated the mean reciprocal rank ($rank^{-1}$) (Voorhees et al., 1999) of d across all questions. As these results closely mirrored the accuracy results, we report accuracy only.

[8] The fraction of the MSR set retained is smaller, because 1000 of the examples in this set concern identifying possessives indicates by the presence of an apostrophe, and terms of this nature were eliminated by the pre-processing procedure.

[9] https://github.com/cambridgeltl/RepEval-2016

performance with the `Semantic Vectors` implementation of the same algorithm. With the random vectors added, out-of-vocabulary rates were equivalent across the two SGNS implementations, resulting in similar performance.

4 Results

4.1 Analogical retrieval

The results of our analogical retrieval experiments are shown in Table 1. With the single exception of the semantic component of the Google set, the best result on every set and subset was obtained by a variant of the $EARP_{prox}$ model, strongly suggesting that (1) information concerning relative position is of value for solving analogical retrieval problems; (2) encoding this information in a flexible manner that preserves the natural relationship of proximity between sliding window positions helps more than encoding only direction, or encoding window positions as disparate "slots".

On the syntactically-oriented subsets (Gsyn, Binf, Bder) adding subword information improves performance of baseline SGNS and EARP models, with subword-sensitive $EARPx_{prox}$ models showing improvements of between $\sim$6% and $\sim$21% in accuracy on these subtasks, as compared with the best performing baseline[10]. The results follow the same pattern at both sliding window radii, aside from a larger decrease in performance of $EARPx$ models on the semantic component of the Google set at radius 2, attributable to semantically useful information lost on account of subsampling without replacement. In general, better performance on syntactic subsets is obtained at radius 2 with subword-sensitive models, with semantic subsets showing the opposite trend.

While better performance on the total Google set has been reported with larger training corpora (Pennington et al., 2014; Mikolov et al., 2017), the best EARP results on the syntactic component of this set surpass those reported from order-insensitive models trained on more comprehensive corpora for multiple iterations. With this subset, the best $EARP_{prox}$ model obtained an accuracy of 76.74% after a single training iteration on a $\sim$9 billion word Wikipedia-derived corpus. Pennington and his colleagues (2014) report a best accuracy of 69.3% after training Glove on a corpus of 42 billion words, and Mikolov and colleagues (2017) report an accuracy of 73% when training a subword-sensitive CBOW model for five iterations across a 630 billion word corpus derived from Common Crawl. The latter performance improved to 82% with pre-processing to tag phrases and position-dependent weighting – modifications that may improve EARP performance also, as would almost certainly be the case with multiple training iterations across a much larger corpus.

Regarding the performance of other order-sensitive approaches on this subset, Trask and his colleagues (2015) report a 1.41% to 3.07% increase in absolute accuracy over a standard CBOW baseline with PENN, and Mikolov and his colleagues (2017) report a 4% increase over a subword-sensitive CBOW model with incorporation of position-dependent weights[11]. By comparison, $EARPx_{prox}$ yields improvements of up to 4.27% over the best baseline when subwords are not considered, and 8.29% with subword-sensitive models (both at radius 5).

With the more challenging BATS analogies, Drozd and his colleagues (2016) report results for models trained on a 6 billion word corpus derived from the Wikipedia and several other resources, with best results with the vector offset method for BATS components of 61%, 11.2% (both SGNS), 10.9% and 31.5% (both Glove) for the Inflectional, Derivational, Lexicography and Encyclopedia components respectively. While not strictly comparable on account of our training on Wikipedia alone and acknowledging only one of a set of possible correct answers in some cases, our best results for these sets were 71.56%, 44.30%, 10.07% and 36.52% respectively, with a fourfold increase in performance on the derivational component. These results further support the hypothesis that order-related information is of value in several classes of proportional analogy problem.

4.2 Semantic similarity/relatedness

These improvements in analogy retrieval were not accompanied by better correlation with human es-

[10]To assess reproducibility, we repeated the window radius 2 experiments a further 4 times, with different stochastic initialization of network weights. Performance was remarkably consistent, with a standard error of the mean accuracy on the MSR, Google and BATS sets at or below .24% (.0024), .33% (.0033) and .12% (.0012) for all models. All differences in performance from the baseline (only $SGNS_{semVec}$ was repeated) were statistically significant by unpaired t-test, aside from the results of $EARP_{dir}$ and $EARP_{prox}$ on the Google set when no subwords were used.

[11]CBOW baselines were around 10% higher than our SGNS baselines, attributable to differences in corpus size, composition and preprocessing; and perhaps architecture.

Radius 2	SW	BATS	MSR	Google	Gsem	Gsyn	Binf	Bder	Blex	Benc
$SGNS_{fastText}$		27.22	53.78	70.13	**77.30**	64.17	56.00	11.76	7.06	32.12
$SGNS_{semVec}$		27.46	53.72	69.53	**77.30**	63.07	56.28	11.65	7.25	32.66
$EARP_{dir}$		28.26	54.48	69.34	76.57	63.33	56.33	11.51	8.36	34.62
$EARP_{pos}$		28.80	54.38	67.71	71.85	64.28	57.22	12.75	8.47	34.70
$EARP_{prox}$		28.95	55.08	69.19	74.95	64.40	57.38	12.86	8.44	35.04
$EARPx_{pos}$		30.50	60.15	61.71	58.94	64.01	66.09	11.07	9.37	33.05
$EARPx_{prox}$		30.79	62.45	66.86	68.80	65.25	66.90	10.76	**9.46**	**33.56**
$SGNS_{fastText}$	✓	28.16	59.83	63.57	57.87	68.31	61.05	22.51	4.47	24.60
$SGNS_{semVec}$	✓	29.17	61.44	65.24	59.38	70.10	62.28	23.56	5.02	25.83
$EARP_{dir}$	✓	31.70	62.94	68.90	62.46	74.26	63.85	29.26	5.54	28.65
$EARP_{pos}$	✓	33.18	62.89	69.35	62.98	74.63	65.10	32.23	6.01	30.12
$EARP_{prox}$	✓	33.26	64.26	**70.82**	64.83	75.80	66.08	31.72	6.05	29.85
$EARPx_{pos}$	✓	36.89	68.90	61.12	44.05	75.29	71.32	44.14	6.81	27.62
$EARPx_{prox}$	✓	**37.07**	**69.07**	63.88	49.31	**75.98**	**71.56**	**44.30**	6.92	27.81

Radius 5	SW	BATS	MSR	Google	Gsem	Gsyn	Binf	Bder	Blex	Benc
$SGNS_{fastText}$		25.54	48.21	69.29	78.47	61.66	49.93	11.95	7.00	31.51
$SGNS_{semVec}$		25.76	46.90	68.75	**79.04**	60.20	49.42	11.84	6.87	33.08
$EARP_{dir}$		27.07	49.20	68.66	76.85	61.85	50.58	12.38	8.30	35.04
$EARP_{pos}$		26.14	45.39	61.29	65.53	57.76	49.82	12.41	7.71	32.80
$EARP_{prox}$		28.40	51.06	69.17	77.05	62.62	53.59	14.05	8.51	35.56
$EARPx_{pos}$		28.30	52.67	59.18	59.80	58.67	57.27	12.90	8.78	32.35
$EARPx_{prox}$		30.94	58.15	69.37	73.51	65.93	59.86	15.36	**10.07**	**36.52**
$SGNS_{fastText}$	✓	26.27	55.04	66.32	64.56	67.79	56.23	18.56	4.43	25.38
$SGNS_{semVec}$	✓	27.69	55.93	67.96	67.37	68.44	57.90	20.32	5.12	26.98
$EARP_{dir}$	✓	30.12	58.20	70.22	69.42	70.88	59.98	24.46	5.78	30.11
$EARP_{pos}$	✓	31.02	54.78	65.67	60.41	70.03	60.28	28.21	5.96	29.94
$EARP_{prox}$	✓	32.67	60.77	**72.23**	69.79	74.27	64.01	28.71	6.46	31.67
$EARPx_{pos}$	✓	34.75	61.08	63.83	54.41	71.66	66.02	36.32	7.51	30.33
$EARPx_{prox}$	✓	**36.67**	**67.44**	72.22	66.78	**76.74**	**69.16**	**38.72**	7.56	32.52

Table 1: Analogical retrieval results at radius 2 (top) and 5 (bottom). SW: subwords. Gsem/syn: "semantic" and "syntactic" components of Google set. Binf/der/lex/enc: inflectional, derivational, lexical and encyclopedia-derived components of the BATS. SGNS: `fastText` and `Semantic Vectors` (*semVec*) implementations of skipgram-with-negative-sampling. EARP: Embeddings Augmented by Random Permutations. EARPx: EARP with exact window positions. Best results are in **boldface**, and rows concerning baseline models are shaded.

timates of semantic similarity and relatedness. To the contrary with a window radius of 2, the best results on all sets and subsets were produced by SGNS baselines, with a best baseline Spearman Rho of .41 ($SGNS_{fastText}$), .72 ($SGNS_{semVec}$) and .77 ($SGNS_{semVec}$ + subwords) for SimLex-999, WS-353 and MEN respectively. Surprisingly, incorporating order-related information reduced performance on all of these sets, with best performance for SimLex-999, WS-353 and MEN of .40 ($EARPx_{prox}$), .71 ($EARP_{pos}$) and .76 ($EARP_{dir}$ + subwords) respectively; and worst performance of .37 ($EARP_{pos}$), .66 and .71 (both $EARPx_{prox}$) respectively. Other models fell between these extremes, with a similar pattern observed with window radius of 5. The differences in performance observed with these tasks are neither as stark nor as consistent as those with analogy tasks, with EARP performance falling between that of the two baseline models in several cases. Nonetheless, EARP models tend to correlate worse with human estimates of pairwise similarity and relatedness. This drop in performance may relate to how semantic information is dispersed with position-aware models - a neighboring word is encoded differently depending on position, which may obscure the semantically useful information that two other words both occur in proximity to it.

Task	CoNLL00	CoNLL03	PTB	CoNLL00	CoNLL03	PTB
$SGNS_{fastText}$	87.09	80.32	94.38	87.34	80.66	94.78
$SGNS_{semVec}$	87.11	77.93	95.59	87.57	80.33	95.96
$EARP_{dir}$	88.25	80.24	95.83	88.06	**80.96**	96.07
$EARP_{pos}$	88.71	80.10	95.74	88.55	80.84	95.89
$EARP_{prox}$	88.59	79.64	95.85	88.78	80.54	**96.09**
$EARPx_{pos}$	**89.05**	80.43	95.92	**89.52**	80.17	95.93
$EARPx_{prox}$	88.99	**80.77**	**96.02**	89.37	79.77	95.93
Subwords	-	-	-	✓	✓	✓

Table 2: Performance on Sequence Labeling Tasks. % accuracy shown for PTB, and % F-measure otherwise

4.3 Sequence labeling

Results of these experiments are shown in Table 2, and suggest an advantage for models encoding position in sequence labeling tasks. In particular, for the sentence chunking shared task (CoNLL00), the best results obtained with an order-aware model ($EARPx_{pos}$ + subwords) exceeds the best baseline result by around 2%, with smaller improvements in performance on the other two tasks, including a .43% improvement in accuracy for part-of-speech tagging (PTB, without subwords) that is comparable to the .37% improvement over a SGNS baseline reported on this dataset by Ling and his colleagues (2015) when using separate (rather than shared) position-dependent context weights.

4.4 Computational performance

All models were trained on a single multi-core machine. In general, `Semantic Vectors` takes slightly longer to run than `fastText`, which takes around an hour to generate models including building of the dictionary. With `Semantic Vectors`, models were generated in around 1h20 when the corpus was indexed at document level, which is desirable as this package uses `Lucene`[12] for tokenization and indexing. As the accuracy of `fastText` on analogy completion tasks dropped considerably when we attempted to train it on unsegmented documents, we adapted `Semantic Vectors` to treat each line of input as an individual document. As this approximately doubled the time required to generate each model, we would not recommend this other than for the purpose of comparative evaluation. Adding subword embeddings increased training time by three- to four-fold. Source code is available via GitHub[13], and embeddings are publicly available at [14].

4.5 Limitations and future work

While this paper focused on encoding positional information, EARP is generalizable and could be used to encode other sorts of relational information also. An interesting direction for future work may involve using EARP to encode the nature of semantic and dependency relations, as has been done with RI previously (Cohen et al., 2009; Basile et al., 2011). As the main focus of the current paper was on comparative evaluation across models with identical hyper-parameters, we have yet to formally evaluate the extent to which hyper-parameter settings (such as dimensionality) may affect performance, and it seems likely that hyper-parameters that would further accentuate EARP performance remain to be identified.

4.6 Conclusion

This paper describes EARP, a novel method through which word order can be encoded into neural word embedding representations. Of note, this additional information is encoded without the need for additional computation, and space requirements are practically identical to those of baseline models. Upon evaluation, encoding word order results in substantive improvements in performance across multiple analogical retrieval reference sets, with best performance when order information is encoded using a novel permutation-based method of positional binding.

Acknowledgments

This work was supported by US National Library of Medicine grant (R01 LM011563), and conducted in part at the University of Texas School of Biomedical Informatics at Houston.

[12]https://lucene.apache.org/

[13]https://github.com/semanticvectors/semanticvectors

[14]https://doi.org/10.5281/zenodo.1345333

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2011. Encoding syntactic dependencies by vector permutation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 43–51. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 1–6.

Trevor Cohen, Roger W Schvaneveldt, and Thomas C Rindflesch. 2009. Predication-based semantic indexing: Permutations as a means to encode predications in semantic space. In *AMIA Annual Symposium Proceedings*, volume 2009, page 114. American Medical Informatics Association.

Trevor Cohen, Dominic Widdows, Manuel Wahle, and Roger Schvaneveldt. 2013. Orthogonality and orthography: introducing measured distance into semantic space. In *International Symposium on Quantum Interaction*, pages 34–46. Springer.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Gregory Finley, Stephanie Farmer, and Serguei Pakhomov. 2017. What analogies reveal about word vectors and their compositionality. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 1–11.

Stephen I Gallant and Phil Culliton. 2016. Positional binding with distributed representations. In *Image, Vision and Computing (ICIVC), International Conference on*, pages 108–113. IEEE.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

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

Michael N Jones, Walter Kintsch, and Douglas JK Mewhort. 2006. High-dimensional semantic space accounts of priming. *Journal of memory and language*, 55(4):534–552.

Pentii Kanerva, Jan Kristoferson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 22.

Pentti Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159.

Jussi Karlgren and Magnus Sahlgren. 2001. From words to understanding. *Foundations of Real-World Intelligence*, pages 294–308.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition. *Psychological Review*, 104(2):211–240.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208.

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

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. 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.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Tony A Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural networks*, 6(3):623–641.

Gabriel Recchia, Michael Jones, Magnus Sahlgren, and Pentti Kanerva. 2010. Encoding sequential information in vector space models of semantics: Comparing holographic reduced representation and random permutation. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Magnus Sahlgren, Anders Holst, and Pentti Kanerva. 2008. Permutations as a means to encode order in word space.

Hinrich Schütze. 1993. Word space. In *Advances in neural information processing systems*, pages 895–902.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neural word embeddings at scale. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 2266–2275. JMLR. org.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.