pre-print version
International Journal of Semantic Computing (IJSC)
Vol: 15 Iss: 02 pppp. 263 - 290
https://dx.doi.org/10.1142/S1793351X21500057?
utm_source=wspc_mail_list&utm_medium=email&utm_campaign=repr_IJSC_140721

Syntactic Coherence in Word Embedding Spaces

Renjith P Ravindran[†] *and Kavi Narayana Murthy*[‡]
School of Computer and Information Sciences,
University of Hyderabad, Hyderabad, India
[†]*rpr@uohyd.ac.in,* [‡]*knmuh@yahoo.com*

Abstract

Word embeddings have recently become a vital part of many Natural Language Processing (NLP) systems. Word embeddings are a suite of techniques that represent words in a language as vectors in an n-dimensional real space that has been shown to encode a significant amount of syntactic and semantic information. When used in NLP systems, these representations have resulted in improved performance across a wide range of NLP tasks. However, it is not clear how syntactic properties interact with the more widely studied semantic properties of words. Or what the main factors in the modeling formulation are that encourages embedding spaces to pick up more of syntactic behavior as opposed to semantic behavior of words. We investigate several aspects of word embedding spaces and modeling assumptions that maximize *syntactic coherence* — the degree to which words with similar syntactic properties form distinct neighbourhoods in the embedding space. We do so in order to understand which of the existing models maximize syntactic coherence making it a more reliable source for extracting syntactic category (POS) information. Our analysis shows that syntactic coherence of S-CODE is superior to the other more popular and more recent embedding techniques such as Word2vec, fastText, GloVe, and LexVec, when measured under compatible parameter settings. Our investigation also gives deeper insights into the geometry of the embedding space with respect to syntactic coherence, and how this is influenced by context size, frequency of words, and dimensionality of the embedding space.

Keywords: word embeddings, syntactic coherence, syntax-semantics interface, pos-tagging

1 Introduction

The Word2vec [45, 47, 48] model has been highly influential in the NLP community, as it has convincingly showed that a remarkable amount of linguistic knowledge implicit in textual data can easily be extracted and encoded as distributed and continuous representation of words known popularly as word embeddings. Word embeddings are said to be distributed, as the representation of a word depends on the representation of other words, and continuous, since the words are represented using continuous numerical values. The success of Word2vec gave rise to other widely used word embedding models such as GloVe [55] and fastText [7]. These algorithms, are in many ways an extension of similar ideas first explored in techniques such as Latent Semantic Analysis [15, 33]. Embedding algorithms process word co-occurrence information in a large text corpus, either implicitly via prediction using Artificial Neural Networks (ANNs) or explicitly through count based statistical models, to

embed words as points, or equivalently, as vectors from the origin to these points, in an n-dimensional Euclidean space having an emergent property that similar words tend to occupy similar spatial positions. Thus word similarity, and hence word meaning, is implicit in the representation. Here meaning of a word is expressed in relation to other words that share its context. This is known as distributional semantics [27] - *words with similar contexts tend to have similar meanings*. Context of a word is defined as the set of words that co-occur with it within a specified window, in the sentences in a given corpus. The idea that word meaning is a function of context is in contrast to traditional sources of word meaning such as dictionaries. With meaning encoded within the representation, word embeddings have been able to impart NLP systems with an intrinsic notion of linguistic knowledge which was otherwise only possible extrinsically via handcrafted rules and separate knowledge resources such as WordNet [49]. Apart from being used as good representation of words in NLP systems, word embeddings have also been seen as a stand-alone linguistic resource. Word embeddings have been used to induce bilingual lexicons by aligning separately learned word embedding spaces from two different languages [46, 24], to study diachronic evolution of languages to measure semantic change over time [26, 58], etc. Another major direction of research interest has been to understand the nature of information encoded in the space and its mechanisms [39, 3, 50].

Capacity of word embeddings to capture both syntactic and semantic aspects of world knowledge has been mainly showcased by the ability to solve word analogy problems such as of the form – *if A is to B then C is to what?* eg. syntactic - `run:ran::sleep:slept`, semantic - `Japan:Tokyo::China:Beijing`. Word embeddings are able to predict the appropriate answer word, the underlined word in the above analogy expressions, because the relation between two words is captured by the vector difference between those words, which in ideal conditions, is expected to be constant for any pair of words sharing that relation. While this ability does not seem to correlate well with performance in specific NLP tasks such as Named Entity Recognition or Sentiment Classification [64, 11], and has been subject to criticisms [41], it nonetheless has become a benchmark test for judging embedding quality. Word analogy problems and other evaluation measures show that both syntactic and semantic information available in textual data can be captured by the embedding algorithm as functions of word context. However, all this opens up a few pertinent questions – *Are syntactic and semantic aspects interleaved and tightly bound to each other in the embedding space? Or are they separate functions of word context, can they be separated out? Are there major factors in the modeling formulation that encourage embeddings to pick up more of syntactic information than semantics? if so, can the existing embedding algorithms be used to extract syntactic word category information?* Unsupervised learning techniques that can induce syntactic category or parts-of-speech (POS) of words are of great value especially for NLP systems in resource poor languages where good quality POS tagged text data is not widely available. If word embedding spaces exhibit high degree of syntactic coherence, ie. if words with similar syntactic properties are located in distinct neighbourhoods in the embedding space, then the words may perhaps be automatically classified to different syntactic categories by cluster analysis of the embedding space.

In this work we probe word embedding spaces for syntactic coherence. Syntactic coherence is the degree to which words in a neighbourhood in the embedding space are of similar syntactic category. In this paper we restrict ourselves to understanding the geometry of word embeddings spaces with respect to syntactic coherence, we do not attempt to perform POS induction via clustering. We use the 100-Million-word POS tagged British National Corpus (BNC) [36] to learn embedding spaces using Word2vec, fastText, GloVe, LexVec [63]

and S-CODE [43] approaches. We use the POS tag information in the corpus to judge syntactic coherence of the learned spaces. Our results show that modeling assumptions made by S-CODE maximize syntactic coherence and thus S-CODE is superior to other models in this regard. We show that syntactic coherence is related to semantic similarity, by analysing the effect of word context and show that syntactic coherence is maximized with a narrow context window and an appropriate embedding dimension. Further, we study the impact of word frequency on syntactic coherence and the distributional behavior of the four major syntactic word categories. More importantly, we critically analyse the major factors in the formulation of word embedding models that tend to maximize syntactic coherence.

2 Related Works

As far as we know there has been no detailed analysis on syntactic coherence of word embedding spaces. However the idea of using word context to infer syntactic categories have been well explored in linguistics, cognitive science and machine learning (ML) literature. In contemporary linguistics, syntactic categories of words are defined by the role a word plays in the structure of a sentence. Syntactic categories therefore can be identified using distributional tests according to which, if a word in a particular position in a sentence can be replaced by another word then the two words belong to the same category. Thus the set of words that can fill in a position (word-slot) in a sentence belong to the same syntactic category. For example all the words that fill the blank in 'The big ____ has fleas.' belong to *noun* category [56]. The distribution of a word's position in the structured environment of a sentence is therefore indicative of its syntactic category. The distributional criteria overcomes the pitfalls associated with notional or semantic criteria of traditional grammar, which try to define syntactic categories using semantics; *nouns* are objects or entities, *verbs* are actions, etc. In cognitive science, distributional analysis is used to model syntactic category acquisition, as part of first language learning in children. Cluster analysis on word co-occurrences in child directed speech, show that distributional information is a major source for syntactic category acquisition [10, 57] along with semantics, morphology and phonology. These studies have shown that word co-occurrence statistics are a reliable source of syntactic information, however these are mostly qualitative studies on small scale data.

In the machine learning community, induction of POS tags was one of the earliest research interests within unsupervised techniques for NLP [12]. While supervised POS tagging consistently gave superior results, the promise of unsupervised methods was to extend NLP to resource poor languages. The main technique employed here is the Hidden Markov Model (HMM), which models a sentence as a sequence of words, which is generated from a sequence of latent states. The latent states correspond to syntactic categories. The unsupervised estimation of the optimal state sequence is then carried out using a variant of the Expectation-Maximization (EM) algorithm known as the Baum-Welch algorithm. Viterbi algorithm is then used to find the optimal state sequence. However, HMMs trained using EM have two main issues – the tendency to assign similar number of words to each state (POS), and the need to specify number of states [31]. Bayesian methods were employed to incorporate required sparsity to reflect the skewed distributions of POS categories observed in language data [22, 18]. Bayesian methods also avoided the need to specify the number of POS tags. The possibility of using word embeddings spaces to induce good quality POS tags was shown by S-CODE embeddings [43, 21]. Unlike HMMs, the S-CODE embeddings model the joint distribution of word co-occurrences directly through distance between embeddings without relying on discrete states. Utilizing symmetric (both left and right) context of

words, S-CODE induces an embedding for each word type. This simpler technique outperformed HMMs, even though it ignored the possibility of multiple tags per word type. The authors suggest that this was likely because the HMMs are over parameterized, as it allowed token specific tags, and that it did not consider symmetric context. Similarly, by making the assumption of one-tag-per-word, [35] showed that unsupervised HMM taggers can be simplified, thereby performing better than the more general models. The current state-of-the-art performance in unsupervised POS induction reported by [67] is obtained by augmenting S-CODE embeddings with paradigmatic context information via substitute vectors along with orthographic features of [6] and morphological features given by Morfessor [14]. More recently, invertible neural networks have been used to obtain embedding spaces with better syntactic coherence [28].

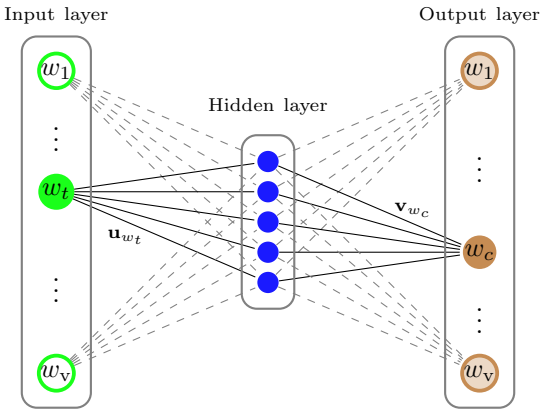


Fig. 1: Word2vec model architecture (simplified).

3 Background

In this section we present the mathematical formulation of word embedding models evaluated in this paper. We bring out the similarities and differences between these models so that we can understand their influence on syntactic coherence. We also briefly mention the existing benchmark evaluations available for word embedding models.

3.1 Word2vec

The Word2vec model [45, 47, 48] is a simple feed forward neural network that is trained to predict word context within sentences. The network with a single hidden layer and certain optimization techniques, makes training even on a Billion word corpus relatively easy. Once training is completed, the weight vector associated with each word in the neural net is taken to be its embedding. Word2vec provides two main variants – Skip-Gram and Continuous Bag of Words (CBOW). In the skip-gram variant, the context prediction is one-to-many, ie. given a target word the network predicts all its surrounding context words. In CBOW the direction is reversed with many-to-one prediction, given a bag of surrounding context words, the goal is to predict the target word. Thus in one variant the target predicts the context and in the other, context predicts the target. The context for every target word in a sentence is decided by the 'window-size' parameter, which allows picking the corresponding

number of words from both sides of the target word in a sentence to form the context. We present below the main aspects of Word2vec with reference to its skip-gram variant. The objective function of skip-gram model is to maximize the following log-likelihood:

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{c \in C_t^s} log\, p(w_c|w_t) \tag{1}$$

$$C_t^s = \{c | t - s \geq c \leq t + s, c \neq t\} \tag{2}$$

Where w_c is a context word of a target word w_t. C_t^s is a set of $2s$ context tokens from either sides of the target token, s is the window size, and T is the total number of tokens in the corpus. Since the likelihood is defined over the entire collection of tokens in the corpus, this formulation implicitly gives more weightage to frequent words. The probability of a word being a context word of a given target word $p(w_c|w_t)$ is defined using the softmax (equation 3), where $\mathbf{u}_w$ is the set of hidden layer weights connecting the corresponding word in the input layer to the hidden layer, and $\mathbf{v}_w$ the weights that connect the hidden layer to the corresponding word in the output layer. Also, V is the size of the vocabulary or the total number of word types in the corpus. Thus each word w has two representations, input representation $\mathbf{u}_w$ and output representation $\mathbf{v}_w$, generally the input representation is taken as final embedding. Figure 1 shows the 3 layer fully connected architecture of Word2vec, where the solid-line connections are the weight vectors ($\mathbf{v}_{w_t}$, $\mathbf{u}_{w_c}$) associated with an active word-context pair w_t, w_c. The softmax relates the conditional probability of a context word given a target word to the dot product between the representations of these words. Since the dot product relates two vectors by their angular distance, higher the probability of two words being in the context of each other, lower will be the angular distance, between their embeddings. The denominator of the softmax is a summation over all words in the vocabulary of the corpus and is computed for every word token in the corpus. Computing the denominator thus becomes impractical for a relatively large training corpus. Word2vec employs techniques such as negative-sampling or hierarchical softmax to solve this issue.

$$p(w_c|w_t) = \frac{e^{\mathbf{u}_{w_t}^\top \mathbf{v}_{w_c}}}{\sum_{j=1}^{V} e^{\mathbf{u}_{w_j}^\top \mathbf{v}_{w_c}}} \tag{3}$$

Negative Sampling [47] scheme is based on Noise Contrastive Estimation (NCE) [25, 52, 51] - a method to fit un-normalized models. The basic idea is to pose density estimation as a classification problem in order to discriminate true samples from noise samples. It uses logistic regression to fit models that are not explicitly normalized thus avoiding the denominator in equation 3. The formulation is as given by equation 4 where $\sigma(x)$ is the sigmoid function used in logistic regression. For each true context word, k negative samples are drawn from a uni-gram[1] distribution N. This approximate likelihood function is a variant of NCE, but it does not share the asymptotic consistency guarantees that NCE offers [16]. However, negative sampling is good enough for representation learning and also gives substantial performance gains.

$$p(w_c|w_t) = \sigma(\mathbf{u}_{w_t}^\top \mathbf{v}_{w_c}) \prod_{w_n \in N_k} \sigma(-\mathbf{u}_{w_n}^\top \mathbf{v}_{w_t}) \tag{4}$$

[1] to be precise, Word2vec uses uni-gram count values raised to the power of $\frac{3}{4}$

Hierarchical Softmax [47], on the other hand, is not an approximation but a more efficient way to compute the Multinomial distribution by leveraging an additional structure. This follows from word-classing in language models [23, 53]. Here the idea is to pose the original Multinomial distribution as product of Bernoulli distributions constructed in a special way using a hierarchical classification of the words. This is done by decomposing the probability mass into a binary tree, where the leaf nodes represent words in the vocabulary and each internal node encodes the probability that the predicted word is in the left/right sub-tree conditioned on its parent node. Unlike the standard formulation where each word gets two representations, here each word gets one representation $\mathbf{u}_w$ and each internal node in the tree gets one representation $\mathbf{v}_n$. The probability of a particular word is expressed as a product of probabilistic binary choices from root node to corresponding leaf node, given by equation 5. $L(w)$ gives the number of nodes in the path from root node to leaf node of word w, $n(w, j)$ is the j^{th} node in path to leaf node of word w, $lc(n)$ gives the left child node of node n, and $[\![a = b]\!]$ results in +1 if $a = b$ else -1.

$$p(w_c|w_t) = \prod_{j=1}^{L(w_c)-1} \sigma\Big([\![n(w_c, j+1) = lc(n(w_c, j))]\!]\mathbf{v}_{n(w_c,j)}^{\mathsf{T}}\mathbf{u}_{w_t}\Big) \tag{5}$$

If a balanced binary tree is used to form the word hierarchy the computational complexity is reduced from $O(V)$ to $O(log_2(V))$, where V is size of the vocabulary. Word2vec uses a Huffman tree and still obtains major computational gains.

3.2 fastText

fastText [7] is a neural word embedding model that is very similar to Word2vec as it borrows Word2vec's overall formulation and optimization techniques. However, it proposes to improve the representation of rare words by incorporating sub-word information. Often rare words are some morphological variants of a frequent word, and thus share some of its sub-word parts. By leveraging sub-word information a rare word may borrow statistical strength from similar words that are more frequent. fastText simplifies the idea of sub-words by simply considering the word itself and all its character-level n-grams, for n in some range, to be its sub-words. Every sub-word is given an embedding and the embedding of a word is taken as the average of embeddings of its sub-words. Word2vec captures distributional interactions between two words via the dot-product of the embeddings of those words. fastText generalizes the interactions to be a function (*score*) that simply sums up the dot-products between sub-word embeddings of corresponding words, given by equation 6, where $G(w)$ gives all the sub-words of word w and $\mathbf{z}_g$ is a corresponding sub-word embedding. The *score* between two words is used to compute the conditional probabilities via the softmax, equation 7. The rest of the formulation is same as Word2vec's.

$$s(w_t, w_c) = \sum_{g \in G(w_t)} \mathbf{z}_g^{\mathsf{T}}\mathbf{v}_{w_c} \tag{6}$$

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{V} e^{s(w_j,w_c)}} \tag{7}$$

3.3 GloVe

Unlike Word2vec and fastText which learn word embeddings by prediction, the GloVe [55] vectors are learned directly from word co-occurrence counts. The *log* of co-occurrence counts for a pair of words within a window across all sentences in the corpus is modeled as the dot-product of two corresponding word vectors in an n-dimensional space. The vectors are then found by weighted least-squares regression model.

$$\mathbb{L} = \sum_{i,j=1,1}^{V,V} f(M_{i,j})(\mathbf{u}_{w_i}^{\intercal}\mathbf{v}_{w_j} + b_i + b_j - log\, M_{i,j})^2 \tag{8}$$

Equation 8 gives the loss function of the GloVe model, which is defined over all word pairs from the vocabulary of size V. $M_{i,j}$ gives the total co-occurrence count of words w_i and w_j in the corpus. $f(M_{i,j})$ weighs each word pair as a function of their co-occurrence count. b_i and b_j are bias terms. The authors claim that GloVe's ability to consider global word statistics is advantageous and therefore better than Word2vec which, during each prediction, has access only to local word statistics.

3.4 LexVec

LexVec [63] is another count based word embedding model that directly process word co-occurrence statistics like GloVe but with a learning procedure that is similar to Word2vec. The co-occurrence statistic utilized by LexVec is the Positive Point-wise Mutual Information (PPMI) and not the log co-occurrence counts as in GloVe. The Point-wise Mutual Information (PMI) is given by equation 9. When a co-occurrence count matrix M with counts of word-context pairs (w_t, w_c) is given, PMI can be computed as in equation 10, where $*$ represents summation across the corresponding index. The PMI is ill-defined for any word pair that does not occur in the corpus, resulting in $-\infty$. In practice, often its variant called Positive PMI is used, equation 11.

$$PMI_{w_t,w_c} = log\frac{p(w_t, w_c)}{p(w_t)p(w_c)} \tag{9}$$

$$= log\frac{M_{w_t,w_c}\, M_{*,*}}{M_{w_t,*}\, M_{*,w_c}} \tag{10}$$

$$PPMI_{w_t,w_c} = max(0, PMI_{w_t,w_c}) \tag{11}$$

It has been shown that the window based negative sampling used in Word2vec performs implicit weighted factorization of a shifted PMI matrix [39]. But unlike other techniques for matrix factorization such as SVD, which uses L_2 reconstruction loss that weighs all errors equally, Word2vec's window based negative sampling leads to a reconstruction loss that weighs error of frequent words more heavily. Since PPMI seems to work better for semantic tasks than PMI [9], LexVec chooses explicit factorization of the PPMI matrix with Word2vec's weighing scheme.

$$\mathbb{L}_{w_t,w_c} = \frac{1}{2}(\mathbf{u}_{w_t}^{\intercal}\mathbf{v}_{w_c} - PPMI_{w_c,w_t})^2 \tag{12}$$

$$\mathbb{L}_{w_t} = \frac{1}{2}\sum_{w_n \in N_k(w_t)} (\mathbf{u}_{w_t}^{\intercal}\mathbf{v}_{w_n} - PPMI_{w_n,w_t})^2 \tag{13}$$

The learning process in LexVec involves minimizing two loss functions. Equation 12 gives the loss function for positive co-occurring word pair tokens w_t, w_c as the L_2 loss between the dot product of their embeddings and the corresponding values in the PPMI matrix. For every such positive pair, k negative words are sampled in a similar way as in Word2vec. Equation 13 gives the loss for such negative samples.

3.5 S-CODE

S-CODE [43] also is a count based statistical model of word co-occurrences. But unlike Word2vec or GloVe which are general purpose word embeddings, S-CODE was introduced as an application of the CODE embeddings to POS induction. S-CODE is the spherical variant of CODE [21]. The CODE embedding is a co-occurrence model for heterogeneous data objects, say images and text, which may be used for exploratory analysis or data visualization. Here, considering word-word bi-grams as data objects, CODE maps a word w to two points in the embedding space, one reflecting its left position in a bi-gram $\phi(w)$ and the other the right position in the bigram $\psi(w)$. Thus the window-size is always 1. The final embedding of a word is the concatenation of the two embeddings. Like other embedding models, the goal is to have the geometry of the embedding space reflect the statistical relation between two words in a corpus. The authors of CODE suggest that, one way to do this is to model the joint distribution $p(x, y)$ of words x and y as proportional to $e^{-d^2_{x,y}}$, where $d^2_{x,y}$ is the Euclidean distance between the embeddings $\phi(x)$ and $\psi(y)$. However, the formulation will have influence of the marginals $p(x)$ and $p(y)$. To model only the statistical relation between x and y avoiding the influence of marginals, CODE considers the ratio $r_p(x, y) = \frac{p(x,y)}{p(x)p(y)}$. The log of $r_p(x, y)$ is the Point-wise Mutual Information (PMI). Equation 14 shows this formulation where Z is the partition function or the normalizing constant.

$$p(x, y) = \frac{1}{Z} p(x)p(y)e^{-d^2_{x,y}} \tag{14}$$

$$Z = \sum_{x,y} p(x)p(y)e^{-d^2_{x,y}} \tag{15}$$

However, this leads to significant complications in parameter estimation as the choice of $p(x), p(y)$ and $d^2_{x,y}$ should all be consistent with $p(x, y)$. The model is simplified replacing the marginals with empirical marginals $\overline{p}(x)$ and $\overline{p}(y)$ as in equation 16.

$$p(x, y) = \frac{1}{Z} \overline{p}(x)\overline{p}(y)e^{-d^2_{x,y}} \tag{16}$$

The likelihood parameterized by the embeddings for words x and y is defined as the expected value of the log model bi-gram probability under the empirical bi-gram distribution (equation 17).

$$\mathcal{L} = \sum_{x,y} \overline{p}(x, y) \, log \, p(x, y) \tag{17}$$

$$= -\sum_{x,y} \overline{p}(x, y) d^2_{x,y} - log \, Z + \sum_{x,y} \overline{p}(x, y) \, log(\overline{p}(x)\overline{p}(y)) \tag{18}$$

Equation 18 can be intuitively understood as follows. The third term in equation is independent of the parameters and is therefore a constant. The first term is the negative mean distance of all points. Due to this term the likelihood is maximized when distances between all points are zero, i.e. when all words are mapped to the same embedding. However the second term acts as a regularizer, pushing embeddings $\phi(x)$ and $\psi(y)$ apart and thus away from the trivial solution.

For a large corpus, computing the normalizing constant Z for each bi-gram in each iteration during the training is impractical. Instead, it is found that by applying a sphere constraint on the embedding space, i.e. by forcing the embeddings to have unit length, the dynamic variable Z can be replaced with constant $\tilde{Z}$ which can be pre-computed. While the sphere constraint stabilizes Z it also makes gradient-ascent considerably smooth allowing for larger step sizes resulting in faster convergence.

3.6 Benchmark Evaluations

The interest in word embedding models is two-fold. One is the interest of the machine learning community to improve performance on various NLP tasks. The other is the interest of cognitive scientists and computational linguists to explain lexical semantics and its acquisition. Similarly, evaluation of word embedding techniques is also of two kinds. **Extrinsic** evaluations test embedding on downstream NLP tasks such as sentiment analysis or machine translation. **Intrinsic** evaluations, on the other hand, test various linguistic capabilities more directly. Major intrinsic evaluation tasks and data-sets can be classified as follows:

1. **Similarity** tasks measure how well the embedding space encodes word similarity the way humans perceive it. Humans are asked to rate given pairs of words for their similarity on a numerical scale. This data-set is then used to measure the correlation (Pearson/Spearman's) between the average human ratings and similarity as measured in the embedding space via cosine similarity. Popular similarity data sets are **RG** [59], **WS-353** [17], **MEN** [8], **SimLex999** [29], etc.

2. **Categorization** tasks require clustering of a set of words into gold standard categories in the embedding space. The purity of the obtained clusters is then measured. Categorization task may be considered as more reliable than the similarity task because choosing a word to belong to a category is relative to other words in the set and not an absolute measure like the similarity scores given by human raters. Popular categorization data-sets are **AP** [2], **Battig** [4], **BLESS** [5], etc.

3. **Analogy** tasks require embeddings to solve word analogy problems. It is posited that the relation between any two words of a given relation is encoded in their corresponding difference vectors. Therefore vector algebra can solve for an unknown vector, eg $\text{Beijing}_{\texttt{vec}} \sim \text{Japan}_{\texttt{vec}} - \text{Tokyo}_{\texttt{vec}} + \text{China}_{\texttt{vec}}$. Solving word analogies were popularized by Word2vec, however the idea was first introduced 40 years before in [60]. The Word2vec papers introduced two data-sets: One with both syntactic and semantic analogies called **Google** and the other with only syntactic analogies called **MSR**.

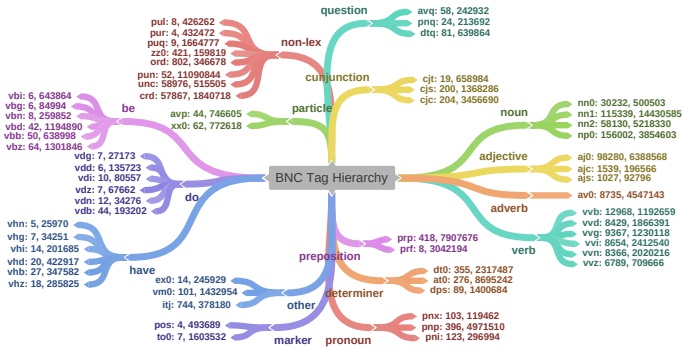


Fig. 2: A hierarchy of BNC tags along with corresponding type and token counts.

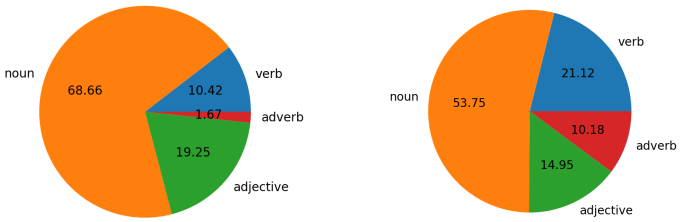
4 Experiment Setup

4.1 Corpus and POS Tag Source

We have used the British National Corpus (BNC) [36], a 100-Million-word corpus of British English in all our experiments. BNC is a synchronic corpus and comprises of a wide range of text samples, both written and spoken (10%), of contemporary British English. The corpus has been annotated with part-of-speech (POS) tags of words using the CLAWS4 tagger [37, 20]. The CLAWS4 tagger is a hybrid multi-pass algorithm with lexicon based, rule based (word endings, templates) and statistical (Viterbi alignment) components. A *core* sample (2 Million words) of the corpus has been manually post-edited. Error analysis on small portions of the corpus showed a tagging accuracy of about 95%. The main tagset (C5) has 57 word-class tags and 4 punctuation tags. In addition to the main tagset an extended tagset (C7) of 152 tags has been used to tag the post-edited *core* sample. Further, the BNC also has what are called portmanteau tags, a total of 30, such as **aj0-nn1**. These tags, also called as ambiguity tags, are assigned to words when the tagger does not have enough information to tell them apart. These portmanteau tags cover nearly 4 million tokens (3851373 to be precise) in the corpus. We do not consider portmanteau tags in our experiments. Figure 2 gives type and token counts for each of the tags (C5). In the figure, the original BNC tags, which we call fine-grained tags, have been grouped into coarse-grained tags forming a hierarchy. For example fine-grained tags such as **nn0** and **nn1** belong to the coarse-grained tag **noun**.

4.1.1 Major and Minor Tags

The BNC tagset has 61 tags in all. We divide the tagset into two sets: Major and Minor tags. Minor tags are the ones that mostly mark functional and closed class words, having low type counts or are non-lexical tokens such as punctuations or numbers. There are 47 of them. For example, forms of *be, have, do* verbs. Word types belonging to these tags are often just a handful but some of these tags have considerable number word types. For example the tag **vbz** is expected to have *is, isn't* as the only two word types. However tokens such as *title-is, family-is* are also tagged as **vbz**. Such hyphenated words make up majority of word types in these tags but these occur only sparingly in the corpus. The **prp**



(a) Word Types (b) Word Tokens

Fig. 3: Proportion of types and tokens in the 4 major coarse-grained tags in BNC.

tag (preposition) is one of the tags that has a good number of word types. Here also 275 of the 418 word types are words that occur less than 10 times in the corpus, in hyphenated form. Another tag that is worth mentioning here that we consider minor is **itj** (interjection). It is generally considered as an open word class, but 569 of 744 word types marked with this tag occur less than 10 times in the corpus.

The remaining 14 tags we consider as **major** tags — **nn0**, **nn1**, **nn2**, **np0**, **vvb**, **vvd**, **vvg**, **vvi**, **vvn**, **vvz**, **aj0**, **ajc**, **ajs** and **av0**. These fine-grained tags are grouped to form coarse-grained tags **noun**, **verb**, **adjective**, and **adverb**, which we shorten to **nn**, **vv**, **aj**, and **av** respectively. We use tag **ot** (other) to refer to the set of all minor tags. Figure 3 gives the percentage of types and tokens belonging to each of the major coarse-grained tags in the BNC.

4.2 Processing of Corpus

For our study we use the current version of BNC - **BNC XML** [13]. The corpus consists of 4049 xml files. We parse each xml file, extract only the required information on to a new single text file. Main considerations in this processing are the following: Any item with a pos-tag is considered a token. However, multi-word-expressions (MWEs) are an exception as these are also given pos-tags. We ignore pos-tag associated with MWEs and consider words within an MWE, each with its own pos-tag, as separate tokens. Each token is then prefixed with its C5 pos-tag (eg: **nn0**:**love**). Word casing is normalised to *lower* and sentence boundaries are maintained.

This gives us a corpus with 111,976,741 tagged tokens. We use this text for all analysis. For creating word embeddings we remove punctuations, numerals and words with brackets and parentheses. We also remove tag-prefixes in each word token giving a token count of 98,359,290 and a type count of 641,794.

4.3 Evaluation Measure: sync-score

POS induction methods that use HMMs implicitly cluster tokens based on their distributional behavior. Quality of induced POS clusters are then evaluated using measures such as many-to-one, V-Measure, etc [12]. Since word embedding models do not form explicit clusters, one needs to perform clustering on the embedding space post the embedding process. However, our goal in this paper is not to perform POS induction. Instead we are only interested in quantifying how well syntactic information is encoded in the word embedding space, without bringing in the complexity and limitations of clustering algorithms. We wish

to measure the degree to which pairs of words within the embedding neighbourhood belong to the same syntactic category. Instead of defining a neighbourhood via clusters we simply look at m nearest neighbours.

$$\text{sync-score} = \frac{\sum_{w \in \mathcal{P}} \sum_{w' \in \mathbb{E}_w^m} \delta_{w,w'}}{|\mathcal{P}| \times m} \tag{19}$$

$$\text{where} \quad \delta_{w,w'} = \begin{cases} 1, & \text{if POS}(w) = \text{POS}(w') \\ 0, & \text{otherwise} \end{cases}$$

This gives us a simple accuracy based measure which we call *syntactic coherence score* or sync-score for short (equation 19). Given a word embedding space, we probe the embedding space with a set of n probe words ($\mathcal{P}$). For each probe w we fetch its m nearest neighbours in the embedding space ($\mathbb{E}_w^m$). For every probe word w and its neighbouring word w' we compute how often w and w' belong to the same syntactic category ie. $POS(w) = POS(w')$. $POS(w)$ returns the common POS tag, either fine-grained or coarse-grain, associated with word w in the given tagged corpus. Thus, sync-score measures the accuracy with which two words in some neighbourhood in the embedding space belong to the same syntactic category. Coherence of word embedding spaces have previously been explored in [9, 64].

4.4 Embedding Model Implementations

For Word2vec and fastText we use the Gensim[2] implementations available for Python. For GloVe and LexVec we use the original implementation of [55][3] and [63][4]. For S-CODE we could not source the original implementation of [43], we use the implementation available in the code distribution of [67][5]. Two major parameters in Word2vec, fastText, GloVe and LexVec are window-size and embedding dimension. While embedding dimension is a major parameter in S-CODE, window-size is not, as it is always 1. All models are trained for 5 iterations, other parameters in respective models are set to defaults in their corresponding implementations. For benchmark evaluations we use the Python module released by the authors of [30][6].

5 Experiments

5.1 Effect of Context on Syntactic and Semantic Behavior

Our first experiment is to evaluate the effect of context size in extracting syntactic behaviour as opposed to semantic behavior. We wish to check if we can clearly understand these as distinct and separate functions of word context. [9] shows that a narrow context (window-size=1) maximize performance on both syntactic and semantic tasks. We however feel that the tasks they employed are not expressive enough to distinguish syntactic properties from semantic properties. Distributional tests from contemporary linguistics may offers some insights. The set of words that can fill a position or word-slot in a sentence should agree

[2] https://radimrehurek.com/gensim
[3] https://nlp.stanford.edu/projects/glove/
[4] https://github.com/alexandres/lexvec
[5] http://www.denizyuret.com/search/label/Downloads
[6] https://github.com/kudkudak/word-embeddings-benchmarks

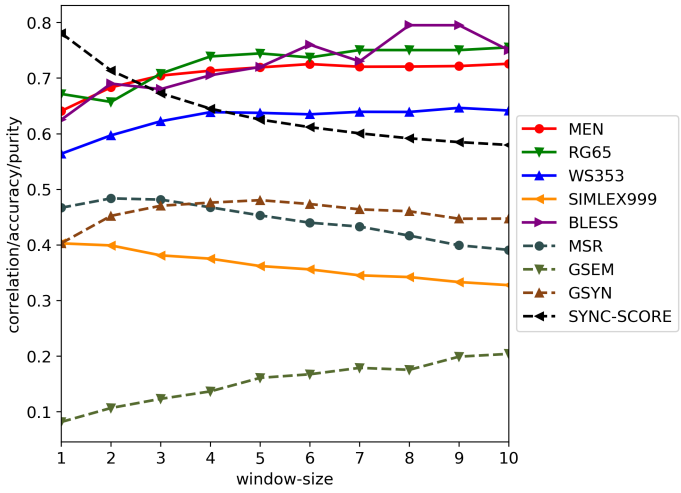


Fig. 4: Results of Exp 1: Effect of window-size on sync-score and other benchmarks.

in their syntactic category. However, at the same time, those words may also agree with the semantics of the sentence. For example the word-slot in "The big ____ has fleas." can be filled by any word that refers to something big that can have fleas, which is a subset of the *noun* category. It is clear that the size of context dictates required semantics, the context here being the full sentence. Instead if we consider a narrower context of one word to the left of the word-slot, the set of words would be a larger subset of category *noun* as the word-slot is influenced only by an *adjective* (big). Anything which can be big may be picked up for filling the slot, there is no further semantic requirements. If we consider a broader context, possibly beyond the given sentence, the set becomes semantically more specific and hence smaller. Therefore, we may argue that broad context windows will capture more of semantics and at the same time also retain appropriate syntactic information.

In distributional semantics the effect of context can be understood in terms of two distinct yet often confounded aspects of semantics – **similarity** and **relatedness**. Two words are similar if they share common attributes, like 'cat' and 'dog', and related if they occur together in the world and in language usage, like 'dog' and 'leash'. It seems there is some consensus that wider context tends to capture more of relatedness and narrow context captures more of similarity [38, 1, 32]. Since syntactic similarity can be seen as the broadest and simplest form of similarity between words, we may reason that syntactic coherence will be maximized with a narrow context. Word similarity does not cross syntactic categories but word relatedness often crosses these boundaries, for eg. 'dog' and 'bark'. Therefore as broader context captures more of relatedness, it can in-effect deteriorate available syntactic information. We put this to test by evaluating the skip-gram model of Word2vec with varying context sizes on syntactic coherence and on major benchmark data-sets for intrinsic evaluation of semantic content. Here we consider words that occur at least 10 times in the corpus to be the vocabulary of the embedding space.

Results and Observations

Figure 4 plots sync-scores for varying window sizes of the skip-gram model along with results of other benchmark tests that showed a consistent dependence on window size. These are MEN, RG65, WS353 and SimLex999 in **similarity**; MSR, semantic and syntactic portion of Google, named GSEM and GSYN respectively, in **analogy**; and BLESS in **categorization**. Since all these measure a value between 0 and 1, we are able show all in one plot. Similarity task measures Spearman's rank correlation, categorization measures cluster purity [42], and analogy measures accuracy. At the outset we see that sync-score degrades rapidly as we increase the window size. It does not stay the same as intuited by the distributional tests. This is consistent with the reasoning we arrived at by relating window size to semantic similarity and relatedness. Looking at the similarity tests, we see that except for SimLex999, increase in window size increases the correlation values. This may seem contrary to our hypothesis, as we expect similarity to degrade with increase in window size. However, the authors of SimLex999 point out that a good portion of word pairs in previous data-sets that have been rated high for similarity are in fact not so similar but more related. SimLex999 was created by explicitly instructing the annotators for measuring similarity instead of relatedness, and therefore this is a stronger benchmark set for similarity. SimLex999 and the other 3 relatedness dominant datasets (RG65, MEN and WS353), therefore are behaving as expected. Coming to tests for analogy, we see that GSEM shows positive correlation with window size. As analogies are expected to capture word relations, word pairs in these datasets exhibit relatedness. Therefore GSEM supports our hypothesis. The trend is somewhat reversed with syntactic analogies as shown by result of MSR. This is because syntactically related pairs, say 'walk' and 'walked', also exhibit good amount of semantic similarity. However, MSR and GSYN are not maximized at window-size=1, but at window-size of 2 and 5 respectively. A possible explanation for this is that analogy tasks measure something more than coherence. In order to solve a word analogy, embeddings of related words should also have a consistent vector difference. This may explain the need for a slightly wider context as it may help stabilize the direction of the difference vector between related word pairs. Lastly the BLESS dataset, in categorization, addresses semantic relatedness, and is consistent with our hypothesis. Thus our experiments generally agree with the hypothesis that wider context windows maximize relatedness of words and narrow contexts maximize similarity [38, 1, 32]. Since syntactic similarity and semantic similarity can not be addressed in complete isolation we see that syntactic coherence is also maximized with narrow context.

5.2 Best Model for Syntactic Coherence

Our second experiment is to find the embedding model that maximizes syntactic coherence. We consider popular general purpose models such as Word2vec, fastText, GloVe and LexVec along with the more specialized S-CODE. We use window-size=1 for all the general purpose models taking cue from the previous experiment. S-CODE by design has a window-size of 1. Since Word2vec has two main variants and two optimization techniques we have four different models. The four Word2vec models are labelled - **sgns**, **sghs**, **bwns**, and **bwhs** for skip-gram with negative sampling, skip-gram with hierarchical softmax, bag-of-words with negative sampling, and bag-of-words with hierarchical softmax respectively. fastText also has these 4 variants, but here we keep only its sgns variant, naming it **fasttext**, as it is the default fastText model. LexVec has 3 variants, a base model as described in section 3.4, a model with positional context [61], and one with sub-word information [62]. Here we keep only the base model, naming it **lexvec**. The GloVe and S-CODE models are

dim	10	24	50	100	300	10	24	50	100	300
model	major fine-grain					major coarse-grain				
sgns	0.59	0.67	0.66	0.65	0.63	0.81	0.82	0.82	0.81	0.79
sghs	0.56	0.66	0.66	0.65	0.62	0.76	0.82	0.82	0.81	0.79
bwns	0.65	0.72	0.72	0.72	0.71	0.84	0.86	0.86	0.86	0.85
bwhs	0.65	0.72	0.72	0.71	0.69	0.84	0.86	0.86	0.86	0.85
fasttext	0.59	0.69	0.69	0.68	0.65	0.80	0.83	0.83	0.82	0.81
glove	0.47	0.49	0.51	0.51	0.51	0.68	0.69	0.70	0.70	0.70
lexvec	0.35	0.47	0.48	0.48	0.45	0.57	0.71	0.71	0.69	0.67
scode	**0.74**	**0.79**	**0.80**	**0.80**	**0.80**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**
rand	0.16	0.16	0.16	0.16	0.16	0.42	0.42	0.42	0.42	0.42

Tab. 1: Results of Exp 2: Sync-scores for embedding models under study.

named **glove** and **scode** respectively. We have seen that distribution of syntactic categories is highly non-uniform (figure 3). For example around 70% of words are *nouns*. To get an idea of maximum sync-score possible by the very nature of word distributions, we test the syntactic coherence of random embedding spaces also. The **rand** embedding models have words attached to random points with the value at each dimension being drawn from a uniform distribution between -2.0 and +2.0. This gives a total of 9 embedding models. Since the optimal embedding dimension for syntactic coherence is not known, all the models are trained on varying embedding dimensions – 10, 24, 50, 100 and 300 giving a total of 45 embedding spaces. The embedding spaces are tested on major fine-grained tags as well as major coarse-grained tags. In this experiment we limit the model vocabulary to words that have occurred at least 100 times in the corpus, giving a vocabulary of around 31K words in each model. All other words in the input corpus are replaced with `UNK` token before training. The syntactic coherence is measured using all 31K words as probe words, and considering 200 nearest neighbours for each word.

Results and Observations

Table 1 gives the results of this experiment. The most striking observation is that S-CODE embeddings are superior to all other embeddings by a large margin, giving fine-grained and coarse-grained sync-scores of 80% and 91% respectively while the next best is given by Word2vec (72% and 86% respectively). The other striking observation is the rather poor performance of GloVe and LexVec embeddings. LexVec mostly performs inferior to GloVe, suggesting that at least when it comes to syntactic coherence, PPMI is no better than bi-gram counts. In Word2vec models, bag-of-words outperforms skip-gram by about 5%. This is due to the fact that skip-gram is a bi-gram model while bag-of-words with window size of 1 is a tri-gram model, and therefore has access to more distributional information at once. Another observation is that of the equivalence of negative-sampling and hierarchical softmax. The former is an approximation of the latter, however negative sampling is almost equivalent or even slightly better than hierarchical softmax. Incorporating sub-word information, as with fastText, offers a small advantage over vanilla **sgns**. Though it should be noted that in this experiment we have dealt only with frequent words, and fastText's claimed advantage is over rare words. Another interesting observation is the influence of embedding dimension on sync-score. The sync-scores seem to peak around dimension 50, though this is not very

freq-range	nn%	vv%	aj%	av%	ot%	total
$[1K, \infty)$	54	20	14	4	7	7195
$[100, 1K)$	60	17	16	3	5	23973
$[80, 100)$	61	15	16	3	5	4225
$[60, 80)$	64	13	15	2	7	6042
$[40, 60)$	64	12	14	2	7	10328
$[20, 40)$	66	10	14	2	8	22747
$[10, 20)$	66	7	15	2	10	33491
$[6, 10)$	64	6	15	1	11	36805
$[1, 6)$	46	3	14	1	21	496771
$[1, 6)^*$	46	3	14	1	20	30000

Tab. 2: Distribution of coarse-grained tags on varying word frequency ranges in the BNC.

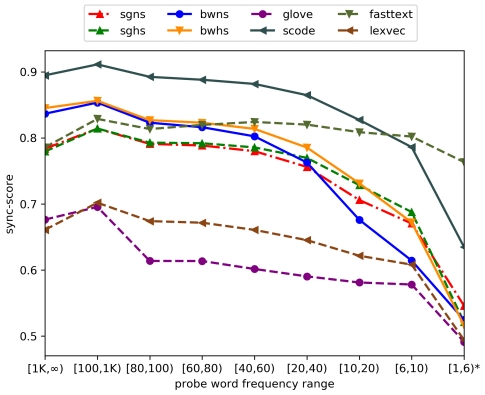


Fig. 5: Results of Exp 3: Sync-scores for varying frequency ranges of probe words.

evident in table 1 due to the decimal rounding. This suggests that lower dimensional spaces may not have enough capacity to encode the required distributional information. Also, after reaching an optimum dimension, adding further dimensions tends to pick up more of noise that degrades the performance. Interestingly we find that some peaks are more flat than others. For example, with S-CODE the maximum for coarse-gained sync-score happens at dimension 100, which is observable only in the 5th decimal place. This suggests that the modeling formulation of S-CODE is not only optimal but also more robust as it is least perturbed by the choice of embedding dimension.

5.3 Syntactic Coherence and Word Frequency

Our third experiment is to throw some light on the behaviour of syntactic coherence as a function of frequency of probe words in the corpus. Frequent words generally obtain good embeddings. Word frequencies in any corpus follow a power-law distribution and have very long tails, accounting for large number of words that occur only a few times. Frequent words are only a tiny minority in any corpus. Table 2 gives the distribution of major coarse-grained tags for words in varying word frequency ranges. Frequency range denoted as $[x, y)$ indicates a set of words that have occurred more than or equal to x and less than y times in the corpus. The frequency range $[1, 6)$ has around half a Million words. To reduce the computational requirements of this experiment we take a random sample of 30K words from the set of words in this frequency range, shown in table 2 as $[1, 6)^*$. For each frequency

range $[x, y)$, a 50 dimensional embedding space is created with words that occur at least x times. This is done with every embedding model under study. For each embedding space with a vocabulary of words that occur at least x times, the set of words in corresponding frequency range of $[x, y)$ is used as probe words to measure major coarse-grained sync-score.

Results and Observations

Figure 5 shows the plot of major coarse-grained sync-score, for all the models having an embedding dimension of 50, against frequency range of probe words. We see that S-CODE embeddings dominate other models on all frequency ranges except the last two. By leveraging sub-word information, fastText is able to beat S-CODE at very low frequency ranges. While performance of all models gradually taper downwards as the frequency of probe words diminishes, fastText is able to maintain its performance, taking a dip only at the lowest frequency range. Incorporating sub-word information clearly stands out as a very effective method to improve representation of rare words. We see that the highest frequency range is not the one with maximum sync-score. This is expected, as most frequently occurring words are also generally highly ambiguous. S-CODE, Word2vec models and LexVec display similar sensitivity towards probe word frequency. However, GloVe's behavior is slightly different. The difference in sync-score from frequency range $[100, 1K)$ to $[80, 100)$ is noticeably larger in GloVe. GloVe appears to be more susceptible to influence of word frequency compared to other embedding models. In the previous experiment negative sampling and hierarchical softmax appeared rather equivalent. But in this experiment we see that when it comes to rare words, as is more evident in bag-of-words models, hierarchical softmax offers a considerable performance boost over negative sampling, giving some credence to the understanding that un-normalized models can approximate the true model when the sample size is large enough, but fail to do so in more parsimonious situations.

5.4 Per Category Syntactic Coherence with Varying Neighborhood Sizes

Our previous experiments explored expected syntactic coherence in the entire embedding space. In this experiment we evaluate the coherence exhibited by words in each of the four major syntactic categories separately. In order to better understand the geometry of the word embedding spaces with respect to syntactic coherence we do so while considering varying number of nearest neighbours in the computation of sync-score. Here we vary the number of nearest neighbours from 50 to 1000 in steps of 100. Note that the number of nearest neighbours is one of the two parameters required to compute sync-score, the other being the set of probe words, variations of which we explored in the previous experiment. We test 50 dimensional embeddings of **sgns**, **bwhs**, **scode**, **fasttext**, **lexvec** and **glove**, with a vocabulary of words occurring at least 100 times. Also, we use the entire vocabulary of the embedding space as probe words.

Results and Observations

Plots in Figure 6 show how per category sync-scores degrade as we increase neighbourhood sizes from 50 to 1000. Across all models we see that *nouns* have relatively high and stable sync-scores up to 1000 nearest neighbours. Coherence of *adverbs* on the other hand is most susceptible to the size of neighborhoods in the embedding space. It is evident that distributional behavior of *adverbs* is least distinct when compared to the rest. However,

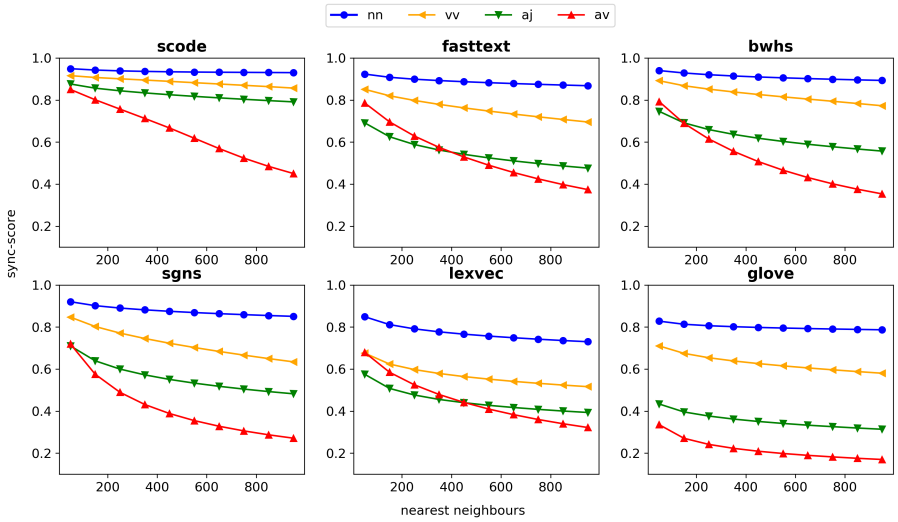


Fig. 6: Results of Exp 4: Syntactic coherence of major syntactic categories with varying neighbourhood sizes.

one must also take in to consideration the fact that while *nouns* are represented here by around 18K words in the embedding space, the number of *adverbs* is only around 1K. Apart from giving high and stable sync-scores for *nouns*, *verbs* and *adjectives* we see that S-CODE's sync-score for *adverbs* is minimally steep over varying neighbourhood sizes. Also interestingly, sync-scores of GloVe seem to be least influenced by larger neighbourhood sizes but the scores are relatively poorer even in small neighbourhoods.

6 Discussion

Our experiments have quantitatively measured the capacity of various word embedding models to encode syntactic category information as distinct neighborhoods. For a more qualitative understanding we visualize the embedding spaces in figure 7, by first performing dimensionality reduction using PCA. The quality of syntactic coherence in S-CODE embeddings stands out from the rest in comparison. It is a clear indicator of the distinct distributional behavior of the four major syntactic word categories, which comes to light under appropriate modeling assumptions. Also, it agrees with the observation we made in the previous section that the distributional behaviour of *adverbs* is least distinct, as these words are observably more spread out in all embedding models.

In each experiment we have tried to reason about specific aspects of the modeling formulation in different embedding models that are indicative of being causative of certain desirable properties when it comes to syntactic coherence. We have only evaluated existing models, and used them as-is. We collate our observations and rationalization below.

1. We clearly see that the optimum context is a symmetric window of size of one, i.e one context word towards the left and one context word to the right of the target word. However, we note that the possibility of asymmetric windows, i.e. either just left context or just the right context, is not explored here. Our study also showed the need to

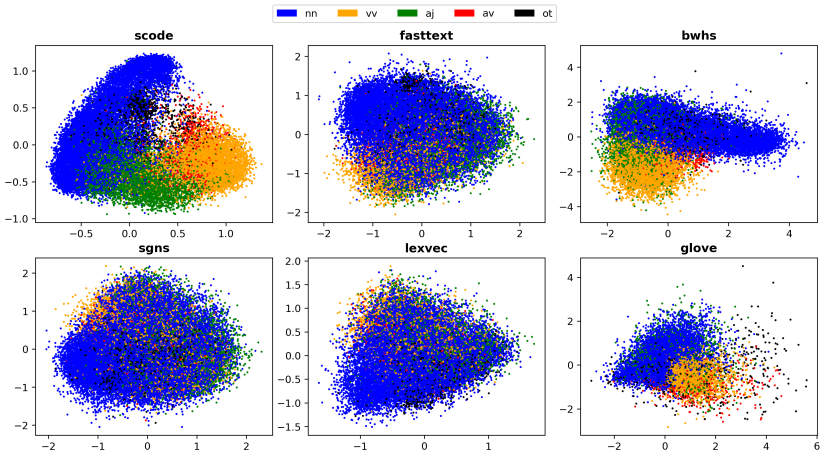


Fig. 7: Visualization of syntactic coherence in the studied embedding spaces.

identify the optimal embedding dimension in order to maximize syntactic coherence. We find that a dimension between 50 and 100 generally works well enough. However, we note that this range is very much dependent on the corpus used, as shown by [68]. But more importantly, they give a theoretical proof that the optimal dimensionality may be understood as the bias-variance trade-off of a novel loss function defined between the real embedding space and the ideal one. This also can be interpreted as the signal-to-noise ratio from information theory, according to which they reason that dimensions below the optimal loses *signal* and the ones above the optimal adds *noise*.

2. We find that S-CODE generally outperforms other models by a large margin. One of the main factors that lead to this is the separation of left and right context in S-CODE's formulation. S-CODE embedding is a concatenation of two separate embeddings, one of left context of words and the other of the right context. In all other models this separation is not maintained. A systematic study in [9] has also shown that keeping contexts separate is better for syntactic tasks than having them summed up.

3. Word2vec introduced two optimization techniques aimed at reducing the training time on a large corpus. In experiment 3 we find that hierarchical softmax is a better alternative, as it handles rare words much better than negative sampling. The two are otherwise fairly equivalent in our case. We could not find any papers in the literature that report similar findings, but this understanding is generally accepted by the authors of Word2vec [44].

4. The optimization trick employed by S-CODE is to constrain the embedding space to be spherical by normalizing the vector norms to unit length. This follows from a finding that doing so stabilizes the value of the normalizing constant which thus can be precomputed, drastically speeding-up the training process. Apart from computational gains, we feel that this is another important factor that contributed to success of S-CODE in our experiments. Because by constraining vector lengths the model is forced to encode information more along the direction of vectors, encouraging distinct vectors to be angularly more spread out. Bounding the max value of the norm, hence

called max-norm, has been understood as a regularization method in collaborative filtering [34]. [19] showed that such norm bounds happen implicitly in sgns due to SGD. [66] showed that explicit normalization improves the retro-fitting of bilingual embeddings used in dictionary induction. Also such normalization is currently part of the 'bag-of-tricks' in deep learning research after being introduced in [65].

5. Incorporating sub-word information into the representation of words came out to be really effective to deal with the problem of sparsity when it comes to rare words. Simply taking the sum of character level n-grams of each word, as shown by fastText, is a good strategy to tackle the problem.

6. A major decision in the modeling formulation is the choice of what to encode in the embedding space. What is the appropriate word co-occurrence statistic that maximizes the signal-to-noise ratio particularly for syntactic coherence? We saw log of bi-gram counts in Glove, Positive PMI in LexVec, and PMI in S-CODE. Word2vec and fastText being predictive neural models, they do not have to choose the appropriate statistic. However, we know that Word2vec's **sgns** implicitly factorizes the PMI matrix [39]. Since S-CODE performs much better than all other models, we may reason that PMI plays a major role. The authors of S-CODE point out that PMI nullifies the influence of marginals (or word uni-gram frequency) therefore is a stronger measure of association than raw joint probabilities (or word-word bi-gram frequency). In our experiments we saw that GloVe was more susceptible to word frequency, suggesting that word frequency information is a source of noise when syntactic coherence is concerned. [64] show that word embeddings can quite accurately predict occurrence frequency of words, GloVe being much better at it than Word2vec. [54] show that top principle components of word embeddings encode word frequency information, thus having those removed makes the embeddings stronger. Therefore we feel that, though Word2vec implicitly deals with PMI, making it explicit as in S-CODE is yet another reason S-CODE trumped over Word2vec.

7. Why does PMI work better than PPMI when it comes to syntactic coherence? PMI may be expressed as $log\ \frac{p(w|c)}{p(w)}$, and takes values from $-\infty$ to $+\infty$. It is the information gained or lost (in bits), about a word when it co-occurs with another word, as measured by its conditional probability, over the information about itself, measured by its marginal probability. PMI is a spectrum of both positive and negative association measures, the negative spectrum being a measure of non-association. To understand why both positive and negative spectrum is important for syntactic coherence than just positive spectrum, we extend an argument made in [39] about why PPMI is more valuable for semantics than PMI. Positive associations are more natural to semantics than negative associations. Humans can easily think of positive associations, say 'snow' and 'Canada', but may find it much harder to come up with negatively associated words, say 'desert' and 'Canada'. But when it comes to syntactic coherence we can easily see otherwise, positive associations tell much about syntactic behaviour of words but so do negative associations. For example, if a word has high negative association say with *adjectives* it is a good indicator that the word may not be a *noun*.

As Word2vec brought vector space models back to prominence, there has been some interest in using it for POS tagging [40, 28]. One should note that word embedding models alone cannot be used for POS tagging. Tagging is generally seen as assigning context specific POS

tags to a sequence of word tokens in a sentence. Word embeddings, of the kind we evaluated here, assume one representation per word type, which is an expectation on all possible distributional behaviors of a word type in the corpus. Therefore, the possibility of token-level POS tagging does not directly fall out of word embeddings. However, the potential for type-level tagging, i.e. assigning a POS tag to each word in a wordlist, is very obvious. POS tag distributions generally tends to be sparse. As a large majority of words in a corpus are likely to take a single predominant tag. Therefore, methods for unsupervised type-level tagging are valuable especially for NLP systems in resource poor languages [35]. Though in this paper we have not explored the effectiveness of type-level tagging via clustering in the embedding space, we hope that syntactic coherence is an appropriate proxy or an important requirement.

Lastly, we have also attempted to study interaction of syntactic coherence with semantic properties of words. Specifically, we have tried to look at syntax and semantics as distinct functions of word context. Our experiments showed that syntax is a function of a narrow word context (window-size=1) and semantics, a function of broad word context. However, we feel it is best to explain both in terms of *similarity* and *relatedness*, because when sync-score is maximized, similarity, as measured by SimLex999, is also maximized. Thus, narrow context essentially captures semantic similarity. POS similarity being the broadest measure of word similarity, it too is maximized. Therefore, we may conclude that syntactic coherence is tied up with semantic similarity which however appears to be largely separate from semantic relatedness.

7 Conclusions

In this paper we have looked at one specific property of word embeddings, namely, ***Syntactic Coherence***. We have shown that a lesser known count based word embedding technique, called S-CODE, generally outperforms the more popular methods such as Word2vec, fastText, GloVe and LexVec in this regard. More importantly, we have critically analysed the various factors in the formulation of word embedding models and how they may influence embedding spaces to have higher syntactic coherence. Our investigations give a deeper insight into the geometry of embedding spaces with respect to syntactic coherence, and how this is influenced by context size, dimension of the space, word frequency and size of neighbourhoods in an embedding space. We have explored these aspects by exhaustive coverage of the word embedding space with a new evaluation measure known as **sync-score** that utilizes POS information in a large corpus. In our attempt to juxtapose syntax and semantics on the grounds of word context, we see that syntactic coherence finds its position at lower rungs in the continuum, *from gross to the subtle*, of semantic similarity, and therefore, may not be completely isolated from semantics. Although our study was limited only to the English language, we believe the insights drawn here should be applicable to other languages as well, and should be of interest to anyone looking for unsupervised POS induction or type-level tagging especially in resource-poor languages.

Acknowledgement

Renjith P Ravindran is funded by Department of Science and Technology (DST), Government of India, under the Inspire Fellowship Programme.

References

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 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, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[2] Abdulrahman Almuhareb and Massimo Poesio. Attribute-based and value-based clustering: An evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 158–165, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[3] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[4] M. Baroni, B. Murphy, E. Barbu, and M. Poesio. Strudel: A corpus-based semantic model based on properties and types. *Cognitive Science*, 34(2):222–254, March 2010.

[5] Marco Baroni and Alessandro Lenci. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK, July 2011. Association for Computational Linguistics.

[6] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California, June 2010. Association for Computational Linguistics.

[7] 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.

[8] Elia Bruni, Nam Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47, January 2014.

[9] John A. Bullinaria and Joseph P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, pages 510–526, 2007.

[10] Timothy A. Cartwright and Michael R. Brent. Syntactic categorization in early language acquisition: formalizing the role of distributional analysis. *Cognition*, 63(2):121 – 170, 1997.

[11] Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 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, Berlin, Germany, August 2016. Association for Computational Linguistics.

[12] Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[13] BNC Consortium. The british national corpus, version 3 (bnc xml edition). Bodleian Libraries, University of Oxford, 2007. http://www.natcorp.ox.ac.uk/.

[14] Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning - Volume 6*, MPL '02, pages 21–30, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[15] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[16] Chris Dyer. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*, 2014.

[17] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January 2002.

[18] Jianfeng Gao and Mark Johnson. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 344–352, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

[19] Yihan Gao, Chao Zhang, Jian Peng, and Aditya Parameswaran. The importance of norm regularization in linear graph embedding: Theoretical analysis and empirical demonstration. *arXiv preprint arXiv:1802.03560*, 2018.

[20] R. Garside. The robust tagging ofunrestricted text: the bnc experience. In J. Thomas and M. Short, editors, *Using corpora for language research: studies in the honour ofGeoffrey Leech Harlow*, pages 167–180. Longman, 1996.

[21] Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean embedding of co-occurrence data. *J. Mach. Learn. Res.*, 8:2265–2295, 2007.

[22] Sharon Goldwater and Tom Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[23] Joshua Goodman. Classes for fast maximum entropy training. *CoRR*, cs.CL/0108006, 2001.

[24] Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 748–756, Lille, France, 07–09 Jul 2015. PMLR.

[25] Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012.

[26] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501. Association for Computational Linguistics, 2016.

[27] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[28] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

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

[30] Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*, 2017.

[31] Mark Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[32] Douwe Kiela and Stephen Clark. A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 21–30, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[33] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240, 1997.

[34] Jason Lee, Benjamin Recht, Ruslan Salakhutdinov, Nathan Srebro, and Joel A. Tropp. Practical large-scale optimization for max-norm regularization. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, page 1297–1305, Red Hook, NY, USA, 2010. Curran Associates Inc.

[35] Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Simple type-level unsupervised pos tagging. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 853–861, 2010.

[36] Geoffrey Leech. 100 million words of english. *English Today*, 9(1):9–15, 1993.

[37] Geoffrey Leech, Roger Garside, and Michael Bryant. CLAWS4: The tagging of the British National Corpus. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*, 1994.

[38] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308. Association for Computational Linguistics, 2014.

[39] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2177–2185, Cambridge, MA, USA, 2014. MIT Press.

[40] Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. Unsupervised POS induction with word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1311–1316, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[41] Tal Linzen. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18. Association for Computational Linguistics, 2016.

[42] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to information retrieval*, chapter Flat Clustering, pages 356–360. Cambridge university press, 2008.

[43] Yariv Maron, Michael Lamar, and Elie Bienenstock. Sphere embedding: An application to part-of-speech induction. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1567–1575. Curran Associates, Inc., 2010.

[44] Tomas Mikolov. Word2vec official web page. https://code.google.com/archive/p/word2vec/. Accessed: 2020-08-30.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[46] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[47] 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, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[48] Tomas Mikolov, Wen-tau Yih, and Geoffrey" Zweig. 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. Association for Computational Linguistics, 2013.

[49] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[50] David Mimno and Laure Thompson. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2873–2878. Association for Computational Linguistics, 2017.

[51] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2265–2273, USA, 2013. Curran Associates Inc.

[52] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 419–426, Madison, WI, USA, 2012. Omnipress.

[53] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252, 2005.

[54] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *CoRR*, abs/1702.01417, 2017.

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

[56] Andrew Radford and SR Anderson. *Transformational grammar: A first course*, volume 1. Cambridge University Press, 1988.

[57] Martin Redington, Nick Chater, and Steven Finch. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive science*, 22(4):425–469, 1998.

[58] Guy D. Rosin, Eytan Adar, and Kira Radinsky. Learning word relatedness over time. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1168–1178. Association for Computational Linguistics, 2017.

[59] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965.

[60] David E Rumelhart and Adele A Abrahamson. A model for analogical reasoning. *Cognitive Psychology*, 5(1):1–28, 1973.

[61] Alexandre Salle, Marco Idiart, and Aline Villavicencio. Enhancing the lexvec distributed word representation model using positional contexts and external memory. *arXiv preprint arXiv:1606.01283*, 2016.

[62] Alexandre Salle and Aline Villavicencio. Incorporating subword information into matrix factorization word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 66–71, New Orleans, June 2018. Association for Computational Linguistics.

[63] Alexandre Salle, Aline Villavicencio, and Marco Idiart. Matrix factorization using window sampling and negative sampling for improved word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 419–424, Berlin, Germany, August 2016. Association for Computational Linguistics.

[64] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[65] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[66] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[67] Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. Learning syntactic categories using paradigmatic representations of word context. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 940–951, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[68] Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 887–898. Curran Associates, Inc., 2018.