Understanding the semantic content of sparse word embeddings using a commonsense knowledge base

Vanda Balogh
University of Szeged
bvanda@inf.u-szeged.hu

Gábor Berend
University of Szeged
MTA-SZTE RGAI, Szeged
berendg@inf.u-szeged.hu

Dimitrios I. Diochnos
University of Oklahoma
diochnos@ou.edu

György Turán
University of Illinois at Chicago
MTA-SZTE RGAI, Szeged
gyt@uic.edu

Abstract

Word embeddings developed into a major NLP tool with broad applicability. Understanding the semantic content of word embeddings remains an important challenge for additional applications. One aspect of this issue is to explore the interpretability of word embeddings. Sparse word embeddings have been proposed as models with improved interpretability. Continuing this line of research, we investigate the extent to which human interpretable semantic concepts emerge along the bases of sparse word representations. In order to have a broad framework for evaluation, we consider three general approaches for constructing sparse word representations, which are then evaluated in multiple ways. We propose a novel methodology to evaluate the semantic content of word embeddings using a commonsense knowledge base, applied here to the sparse case. This methodology is illustrated by two techniques using the ConceptNet knowledge base. The first approach assigns a commonsense concept label to the individual dimensions of the embedding space. The second approach uses a metric, derived by spreading activation, to quantify the coherence of coordinates along the individual axes. We also provide results on the relationship between the two approaches.

1 Introduction

Word embeddings developed into a major tool in NLP applications. An important problem – receiving much attention in the past years – is to study, and possibly improve, the *interpretability* of word embeddings. As interpretability is a many-faceted notion which is hard to formalize, its evaluation can take different forms. One approach is *intrusion detection* [12, 22], where human evaluators test the coherence of groups of words found using word embeddings. A basic observation is that *sparsity* of word embeddings improves interpretability [12, 30].

In order to perform a systematic study, we consider several methods to generate sparse word embeddings from dense ones. One family of word embeddings is obtained by *sparse coding* [5], another by *clustering*, and a third by greedily choosing *almost orthogonal* bases.

Another important problem, also receiving much attention is to combine word embeddings and *knowledge bases*. Such a combination has the potential to improve performance on downstream tasks. The information contained in a knowledge base can be incorporated into a word embedding in different ways either during [18, 24] or after [11, 15] the construction of the word embeddings.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

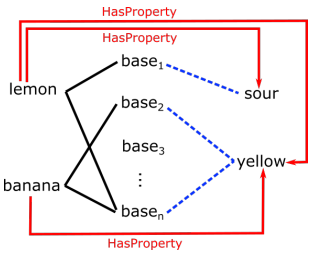


Figure 1: Tripartite graph presenting the connections between embedded words, bases and concepts. Connections indicated by solid lines are initially given, and we are interested in extracting the relationships between bases and commonsense concepts marked by the dashed connections.

A knowledge base provides different tools to explore the semantic content of directions, and thus of the basis vectors (also referred to as *semantic atoms*) in sparse word embeddings. These tools include *concepts* contained in a knowledge base and notions of *semantic relatedness* derived from a knowledge base [13]. The former can be *simple* or *composite*, the latter can be notions based on *graph distances and edge labels, e.g., using spreading activation, label propagation or random walks*.

Knowledge bases give a principled computational approach for the two problems on word embeddings mentioned above (interpretability and knowledge bases), by providing explicit *"meanings"* with quantifiable validity, which capture the implicit coherence of groups of words in general. We focus on *commonsense* knowledge bases, in particular on ConceptNet [29], as commonsense knowledge seems to be a fundamental problem where progress coming from such a combination of statistical and symbolic approaches could be relevant.

In this paper we report recent results on a systematic study of *explicit* connections between word embeddings and knowledge bases. The approach is illustrated by Figure 1.

Section 3 describes three types of sparse word embeddings discussed, and compares them in terms of incoherence and the overlap between word vectors and semantic atoms. Section 4 introduces the algorithm for assigning ConceptNet concepts to bases in word embeddings and the different quantities from information retrieval measuring the quality of the assignments. The experiments performed evaluate the assignments for the three types of embeddings. Section 5 develops the tool for the other evaluation approach: using ConceptNet to measure coherence or semantic relatedness of a set of words by spreading activation. This is then used for experiments evaluating words corresponding to bases in the sparse embeddings. Section 6 brings the two approaches together by analyzing their correspondences.

2 Related work

Faruqui et al. [12] and Subramanian et al. [30] are seminal papers on sparse word embeddings. In particular, Subramanian et al. [30] mention that "sparsity and non-negativity are desirable characteristics of representations, that make them interpretable" as a hypothesis. Investigating this hypothesis using quantitative evaluation is one of the objectives of our paper.

Tsvetkov et al. [32] introduced the evaluation measure QVEC for the quality of a word embedding space. QVEC computes a correlation between the dimensions of the space and the semantic categories obtained from SemCor [21]. QVEC-CCA [31] was introduced as an improvement, relying on canonical correlation analysis [17]. Compared to our paper, both QVEC and QVEC-CCA provide an overall statistical measure rather than an explicit interpretation, and interpretations are given in terms of a relatively small number of lexical categories. QVEC correlates positively with performance on downstream tasks, i.e., more interpretable word embeddings (in the QVEC sense) perform better.

Şenel et al. [28] consider explicit assignments to word embedding dimensions, and propose specific interpretability scores to measure semantic coherence. This is perhaps the paper most closely related to our approach. They introduce a new dataset (SEMCAT) of 6,500 words described with 110 categories as the knowledge base. [28] considers dense word embeddings. In contrast, our paper investigates sparse word embeddings from multiple aspects, and it is based on ConceptNet, which is much larger and richer but also noisier than SEMCAT.

Osborne et al. [24] introduced an algorithm for word representations encoding prior knowledge besides the distributional information. Alsuhaibani et al. [2] consider learning a word embedding and a knowledge base together. The knowledge base is incorporated into the embedding *implicitly* by integrating it into the objective function (i.e., vectors of words in a relation are supposed to be close). Several papers take a similar approach to utilize background knowledge in deep learning, e.g., TransE (Border et al. [7]). In the other direction, similarity of vectors is used for updating the knowledge base. Gardner et al. [14] uses word embeddings similarity to aid finding paths for new relation tuple prediction. Evaluations are typically performed on downstream tasks. Explicit concept assignment – proposed in this paper – could be an additional tool for all these approaches.

Path-based methods for semantic relatedness are surveyed among other methods, e.g., in Feng et al. [13]. Harrington [16] considers spreading activation-based methods in ASKNet semantic networks. Berger-Wolf et al. [6] considers spreading activation in ConceptNet 4 for question answering.

3 Sparse word models

We created sparse word representations based on multiple strategies. Here we introduce the different approaches employed during our experiments.

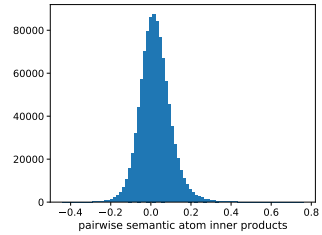
Dictionary learning-based sparse coding (DLSC) The first approach we employed was dictionary learning-based sparse coding (DLSC). DLSC is a traditional technique for decomposing a matrix $X \in \mathbb{R}^{v \times m}$ into the product of a sparse matrix $\alpha \in \mathbb{R}^{v \times k}$ and a dictionary matrix $D \in \mathbb{R}^{k \times m}$, where k denotes the number of basis vectors (semantic atoms) to be employed. In our case X is a matrix of stacked word vectors, the rows of D form an overcomplete set of basis vectors and the sparse nonzero coefficients in the i^{th} row of α indicate which basis vectors from D should be incorporated in the reconstruction of input signal $\mathbf{x}_i$. DLSC optimizes for $\min_{D \in \mathcal{C}, \alpha \in \mathbb{R}^{v \times k}_{\geq 0}} \frac{1}{2}\|X - \alpha D\|_F^2 + \lambda \|\alpha\|_1$, where $\mathcal{C}$ is the convex set of matrices with row norm at most 1 and the coefficients in α has to be non-negative. We imposed the non-negativity constraint as it has been reported to provide increased interpretability [22]. We used the SPAMS library [19] to solve the above optimization problem.

We utilized 300-dimensional Glove embeddings [25] pre-trained on 6 billion tokens. The embeddings consist of the 400,000 most frequent lowercased English words based on a 2014 snapshot of Wikipedia and Gigaword 5. Unless stated otherwise, we set our hyperparameters as $\lambda = 0.5$ and $k = 1000$.

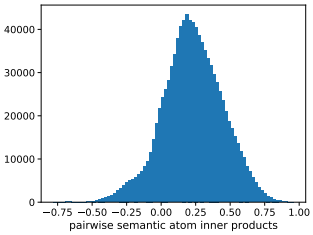
Determining semantic atoms based on clustering As semantic atoms can be also viewed as representative *meta-word vector*s, we also constructed D by performing k-means clustering of the actual word vectors as well. Note that k-means can also be considered as a special case of the k-SVD sparse coding algorithm [1]. We set $k = 1000$ similar to DLSC and determined the semantic atoms comprising D as the cluster representatives, i.e., the centroids of the identified clusters.

Determining almost pairwise orthogonal semantic atoms As the semantic atoms can be regarded as prototype vectors in the original embedding space, we introduced an approach which treats actual word vectors originating from the embedding matrix X as entries of the dictionary matrix D. Since the dictionary learning literature regards the incoherence of dictionary matrices as a desirable property, we defined such a procedure which explicitly tries to optimize to that measure. The proposed algorithm chooses the dense word vector corresponding to the most frequent word from the embedding space as the first vector to be included in D. Then in $k - 1$ subsequent steps, the dictionary matrix gets extended by $\mathbf{x} \in X$ which minimizes the score $\max_{\mathbf{d_i} \in D} |\langle \mathbf{x}, \mathbf{d_i} \rangle|$. We shall refer to this procedure as the **g**reedy **m**aximization for the **p**airwise **o**rthogonality of the semantic atoms, or GMPO for short.

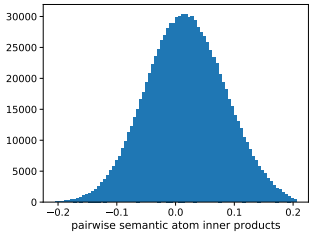
Comparison of the different approaches The formal notion of incoherence [3] gives us a tool to quantitatively measure the diversity of a dictionary matrix $D \in \mathbb{R}^{k \times m}$, according to $\max_{\mathbf{d_i} \neq \mathbf{d_j}} \langle \mathbf{d_i}, \mathbf{d_j} \rangle / \sqrt{k}$, with $\langle \cdot, \cdot \rangle$ denoting the inner product. As incoherence of the dictionary matrix has been reported to be an important aspect in sparse coding, we analyzed D from that perspective. Figure 2a illustrates the pairwise inner products between the semantic atoms from the dictionary matrix D in the case of the DLSC method. We can observe that the semantic atoms are diverse, i.e., the inner products concentrate around zero. From the perspective of incoherence, the dictionary matrix obtained by performing k-means clustering has a lower quality (higher incoherence



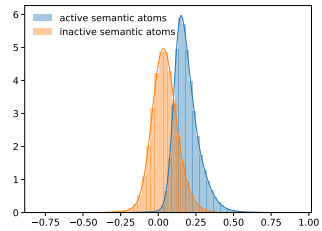
(a) DLSC: Cosine similarities between semantic atoms (Incoherence=0.024)



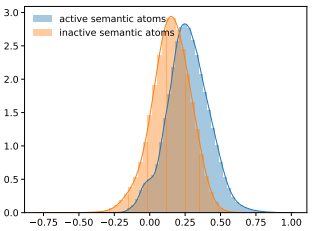
(b) k-means: Cosine similarities between semantic atoms (Incoherence=0.030)



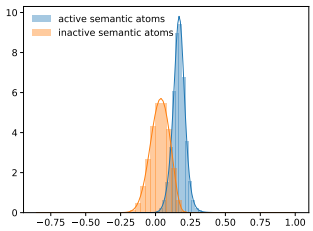
(c) GMPO: Cosine similarities between semantic atoms (Incoherence=0.007)



(d) DLSC: Cosine similarities of word vectors and semantic atoms



(e) k-means: Cosine similarities of word vectors and semantic atoms



(f) GMPO: Cosine similarities of word vectors and semantic atoms

Figure 2: Characteristics of matrices D and α when different approaches are used for determining D.

score) as also illustrated by the pairwise inner products of the semantic atoms in Figure 2b. Figure 2c demonstrates that keeping the pairwise orthogonality of the semantic atoms in mind (cf. GMPO) indeed results in a more favorable incoherence score of 0.007.

We now define active and inactive semantic atoms with respect to some word vector $\mathbf{x_i}$. We say that a semantic atom $\mathbf{d_j}$ is active with respect to $\mathbf{x_i}$, if $\mathbf{d_j}$ takes part in the reconstruction of $\mathbf{x_i}$, i.e., when $\alpha_{ij} > 0$. Additionally, we define the semantic overlap between a semantic atom $\mathbf{d_j}$ and a dense word vector $\mathbf{x_i}$ as $\langle \mathbf{x_i}, \mathbf{d_j} \rangle$, i.e., the projection of $\mathbf{x_i}$ onto $\mathbf{d_j}$. We can see in Figure 2d that the semantic overlap of word vectors towards active semantic atoms tend to be higher than for inactive ones, suggesting that we managed to learn meaningful sparse representations. As semantic atoms are less dissimilar from each other in the case of the k-means approach, we observed that the distribution of the active and inactive (semantic atom, dense word vector) pairs is also less distinguishable from each other (cf. Figure 2e). In accordance with the low incoherence score for GMPO, Figure 2f reveals that the difference in the distribution of the semantic overlap between active and inactive semantic atoms towards the dense input vectors is the most pronounced for GMPO.

We also compared the sparsity levels obtained by the different approaches. Table 1 contains the number of nonzero coefficients on average. The k-means approach produces fewer nonzero coefficients when using the same regularization coefficient ($\lambda = 0.5$). The second row of Table 1 reveals that this comes at the price of performing worse in the reconstruction of the original dense embeddings.

Table 1: The number of nonzero coefficients assigned to a word on average and the total reconstuction error incurred during the reconstruction of the embedding matrix X.

	DLSC	k-means	GMPO
Avg. nnz in α per word	52.86	19.41	59.64
Error term ($\|X - \alpha D\|_F$)	2734.5	3286.9	2971.8

4 Base assignment

Our first approach to investigate the interpretability of the dimensions of sparse embedding matrices is assigning each dimension human interpretable features similar to our previous work [4]. The rows of the embedding matrix correspond to sparse word vectors representing words. We call the columns (dimensions) of the sparse embedding matrix *bases*. As human interpretable features, we take concepts extracted from a semantic knowledge base, ConceptNet.

We focus on the English part of ConceptNet 5.6 [29] which consists of *assertions* that associate pairs of words (or phrases) with a semantically labelled, directed relation. A word (or phrase) in ConceptNet can either be a start node, an end node or both. In our setting, start nodes correspond to *embedded words* and we call the end nodes *concepts*. We keep only those concepts that appear more than 40 times as end nodes in ConceptNet. We use the the 50k most popular words (based on total degrees) in ConceptNet that are also among the embedded words. Basically, we have a tripartite graph (see Figure 1) in which words can be connected to bases and concepts. A word w is connected to $base_i$ if the ith coordinate of the sparse word vector corresponding to w is nonzero. Also, w is connected to a concept c if there exists an assertion in ConceptNet that associates w and c. We are interested in the relations between concepts and bases (dotted lines). In other words, our goal here is to analyze to what extent the sparse embedding is in accordance with the knowledge base.

4.1 Base assignment algorithm

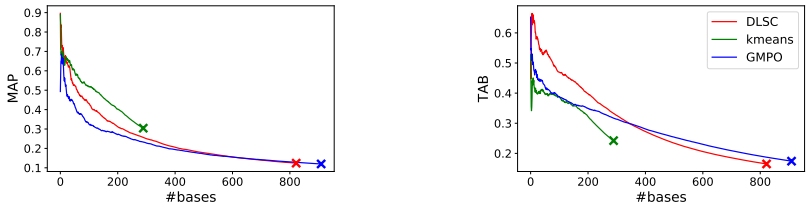
ConceptNet can be viewed as a bipartite graph represented by a matrix C, where an entry $C(w, c)$ is 1 if word w is associated to concept c, and 0 otherwise. We introduce a similar binary matrix B that we obtain from the matrix of sparse coefficients α by replacing the actual coefficients with indicator variables that take the value 1 for non-zero coefficients. We then consider the product $A = C^T B$, the a_{ij} element of which contains the number of times some word assigned to concept i in ConceptNet has base j included in its sparse decomposition. We next derive a matrix containing the normalized positive pointwise mutual information (NPPMI) for every pair of concept c_i and base b_j as $\mathrm{NPPMI}(c_i, b_j) = \max\left(0; \ln \frac{\mathrm{P}(c_i,b_j)}{\mathrm{P}(c_i)\mathrm{P}(b_j)} \Big/ -\ln \mathrm{P}(c_i, b_j)\right)$, with $\mathrm{P}(c_i), \mathrm{P}(b_j), \mathrm{P}(c_i, b_j)$ denoting marginal and joint probabilities for c_i and b_j approximated from matrix A. Finally, we take argmax in every column of the NPPMI matrix to find the concept associated with the bases. If even the highest NPPMI score for some base is zero – implying no positive dependence for that base towards any of the concepts – we do not assign any concept to it.

As a post processing step, we compute NPPMI for concept pairs as well. Alongside the associated concept c of a base b, the concepts that are close to c are also assigned to b, thus creating *meta-concepts*. The set of close concepts for c is $\mathrm{close}(c) = \{c' | \mathrm{NPPMI}(c, c') \geq \max(0.5, 0.95 * \max_{c'' \neq c}(\mathrm{NPPMI}(c, c''))\}$. After the introduction of meta-concepts, 2.55, 2.39 and 2.56 concepts are assigned on average to a base for the DLSC, k-means and GMPO approaches, respectively.

4.2 Evaluation

To evaluate the associations between bases and concepts, we employ information retrieval metrics [20]. We measure if the *dominant words* of a base, i.e., the words for which the given base is active (as defined in Section 3), are in relation with the concepts associated to the base according to ConceptNet.

We use mean average precision (MAP) as a precision oriented metric during our evaluation. MAP is calculated for the first 50 words that have the highest nonzero values for every base. If a base has no concept assigned to it, the average precision and the reciprocal rank of that base is set to zero. As for recall oriented metrics, similarly to [28], train and test words are randomly selected (60%, 40%) for each concept before the assignment takes place. On average each concept has 40 test words. The assignments are obtained from train words (described in Section 4.1), and for each concept its test words are removed. Afterwards, the percentages of unseen test words are calculated in two different ways. The first one measures accuracy of the test words according to bases and it is called *test accuracy by bases* (TAB). Formally, $\mathrm{TAB}(b) = \frac{|\{w \in D_b \cap \mathrm{test}(c)\}|}{|\{w \in V | (w,c) \in KB \wedge w \notin \mathrm{train}(c)\}|}$, where D_b is the set of nonzero coefficient words in base b, c is the concept assigned to base b, V is the set of all words, KB stands for the knowledge base, furthermore $\mathrm{test}(c)$ and $\mathrm{train}(c)$ are the set of test and train words for concept c, respectively. The other metric we use measures *test accuracy by concepts*



(a) MAP (b) TAB

Figure 3: Cumulative evaluation scores for MAPand TAB. The horizontal axis shows bases cumulatively ordered in ascending order with respect to their highest NPPMI values. After the crosses NPPMI values are zero, meaning that no new concept assignment could be performed afterwards.

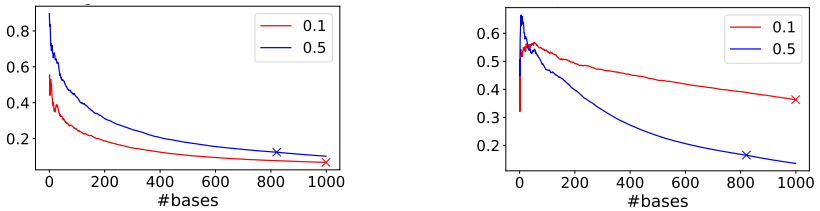
Table 2: Mean and standard deviation of TAC computed for all assigned concepts and the harmonic mean of MAP and TAB.

Approach	Mean$_{TAC}$	Std dev$_{TAC}$	Harmonic mean of MAP and TAB
DLSC	0.498	0.241	0.105
k-means	0.450	0.201	0.072
GMPO	0.497	0.228	0.117

(TAC) and it is calculated for some concept c as $\mathrm{TAC}(c) = \frac{|\{w \in (\cup_b \{D_b | b \text{ has } c \text{ assigned }\}) \cap \mathrm{test}(c)\}|}{|\{w \in V | (w,c) \in KB \wedge w \notin \mathrm{train}(c)\}|}$. The average is taken over all bases for TAB and all concepts in the case of TAC. In order to combine the precision and the recall-oriented views, we compute the harmonic mean of MAP and TAB.

Figure 3 shows the results of MAP and TAB cumulatively. The bases are always in ascending order according to NPPMI values. The evaluation metric with respect to all the bases is always the value at the end of the horizontal axis. Generally (as seen in the monotone behaviour of curves in Figure 3), *the NPPMI values correlate with the evaluation metrics*. As long as k-means has bases that have assigned concepts (shown as a cross in the figures), it performs the best in terms of MAP. However, DLSC and GMPO have a lot more bases that have concepts assigned to them. On the long run, GMPO slightly outperforms DLSC at MAP. Figure 3b and Table 2 reveals that DLSC and GMPO perform similarly and better than the clustering-based approach for the further evaluation metrics.

Next, we investigate a less conservative regularization coefficient, $\lambda = 0.1$. We report its effects for the DLSC approach only for space considerations. The average number of nonzero coefficients per a word increased from 52.9 to 186.9 when using $\lambda = 0.1$ instead of $\lambda = 0.5$. Figure 4 illustrates that sparser representations favor evaluation towards MAP, while TAB performs better in the case of representations with lower sparsity.



(a) MAP (b) TAB

Figure 4: Comparison of evaluation scores on DLSC sparse embedding with different regularization coefficients. Precision related metrics tend to favour sparser solutions (λ=0.5), recall oriented metrics gravitate towards less sparse representations (λ=0.1).

Table 3: Results obtained using spreading activation on ConceptNet 5.6. APL_t and APL_b correspond to the average path length for pairs of top and bottom words, respectively. The last column titled n/a counts bases for which we could not complete the experiments due to memory constraints.

approach		size of activated network				comparing average path lengths			
		min	median	average	max	$APL_t < APL_b$	$APL_t > APL_b$	ties	n/a
DLSC	top smaller	740	661	630	554	657	300	23	20
	bottom smaller	238	319	350	426				
	ties	2	0	0	0				
k-means	top smaller	768	761	703	580	667	299	13	21
	bottom smaller	209	218	276	399				
	ties	2	0	0	0				
GMPO	top smaller	766	685	651	563	731	238	18	13
	bottom smaller	219	302	336	424				
	ties	2	0	0	0				

5 Spreading activation and ConceptNet

Collins and Quillian [9] were the first to show evidence that categories of objects form a hierarchical network in human memory and through this hierarchy meaning could be given to words. Applications for knowledge bases build on such hierarchical structure in order to find semantic similarity between words, semantic relatedness, meaning, as well as for question answering. Among the main tools used in such applications are *label propagation* [26] and *spreading activation* [8]; e.g., [27, 16, 23, 6].

Label propagation methods starting with two nodes having distinct labels, proceed in iterations where a label is propagated to neighbors that obtained the label in the previous round. Ultimately, a node (or a set of nodes) is reached where both starting labels appear. Such nodes are important as they allow the formation of a short path between the two starting nodes without looking at the entire network. Spreading activation methods build on this idea; in each round apart from propagating labels, activation values are propagated along the relations connecting the various words, allowing additional filtering so that *heavy* short paths are found connecting the starting nodes.

We employ spreading activation in ConceptNet 5.6 to investigate the coherence of dominant words in each base. Now we allow non-English words to be activated as well. Such an example is given at the end of the current section. We are interested if the dominant words in a base make a semantically coherent group compared to the words with zero coefficients. With this goal in mind, 10 words with the largest nonzero coefficients are selected from each base (if possible) and also, 10 words with zero coordinates are randomly chosen. We call these two sets of words *top* and *bottom* words of a base, which always come from the 50k highest total degree words in ConceptNet.

Table 3 presents findings from our experiments. For the paths found, the average path length among pairs of top words (APL_t) is less than the average path length among pairs of bottom words (APL_b) in about $66\% - 73\%$ of the bases. Interestingly, *the network activated while searching for a path is typically smaller for pairs of top words compared to the one obtained for pairs of bottom words.*

On the average path lengths When APL_t has a value of *3.044 or less* then that value is always smaller than APL_b. This is true for *all three algorithms*. Furthermore, when APL_t has a value of about *2.5 or less*, then such words are very well aligned and all of them are typically members of a broader group. As APL_t increases, the coherence among the top words fades out. Table 4 in Section 6 provides some examples; Appendix D has further evidence.

On the spreading activation variant The spreading activation variant we use behaves similarly to label propagation. In almost all cases the path connecting a pair of words is one of the shortest found in the knowledge base and the activation helps us identify a heavy such short path. This approach is in accordance to our basic intuition that words that have good alignment with particular bases should form coherent groups and we would expect this coherence to be exemplified by short paths connecting such pairs of words. Appendix has more information on the method used.

On the alignment In some cases the top 10 aligned words with a particular base do not form a (very) coherent group. For example, with the DLSC dictionary, in base 609, the top words are: contiguity, plume, maghreb, tchaikovsky, acuminate, maglev, trnava, interminably, snowboarder, and convalesce. In fact this is an example where the top words have average path length *more than* that of the bottom words (4.044 vs 3.644); so the incoherence of the top aligned words is reflected in the path lengths.

Table 4: Coherent top words in some bases of the DLSC embedding and the assigned concepts. APL_t and APL_b show the average path length for the top 10 and bottom 10 words, respectively.

Concepts assigned	Top words	APL_t	APL_b
china, prefecture	china changchun chongqing tianjin wuhan liaoning xinjiang shenyang shenzhen nanjing	1.84	3.40
farm, farmer	maize crops wheat grain crop soybean sugarcane corn livestock cotton	1.87	3.96
drug, pharmaceutical drug	antidepressant drug tamoxifen drugs statin painkiller aspirin stimulant antiviral estrogen	2.00	4.07
death, funeral, die	slaying murder stabbing murdering death beheading killing murderer hanged manslaughter	1.96	3.58
payment, pay	payment deductible expenses taxes pay pension refund tax tuition money	1.73	3.40

Table 5: Pearson correlations (ρ) between the assignment evaluations (MAP, TAB) and the average path length of top words for sparse word models. We report p-values for the ρ in parenthesis.

	DLSC	k-means	GMPO
ρ_{MAP}	-0.60 (1.1e-98)	-0.58 (6.1e-88)	-0.53 (3.2e-73)
ρ_{TAB}	-0.60 (3.0e-97)	-0.59 (8.0e-93)	-0.53 (1.3e-62)

On polysemy In several cases it is the phenomenon of polysemy that gives the path which is short and heavy. This issue can happen when looking at paths for both top and bottom words and regardless of the overall coherence of the words in the group. For example, when using the k-means dictionary, for base 48, the top words *trad* and *volcanologist* are found to be connected with the path: /c/en/trad – /c/en/music – /c/en/rock – /c/fr/géologie – /c/en/volcanology – /c/en/volcanologist.

6 Discussion and synthesis of results

Now we synthesize the evaluation of base assignments with coherence analysis. The 50k highest degree words in ConceptNet are used. Qualitative results are in accordance with quantitative ones.

Generally, *concepts that were assigned to bases reflect their dominant words*. Table 4 shows bases where average path length among dominant words was much lower than among non-dominant ones, which implies coherence of the base. Clearly, there is a strong connection between assigned concepts, dominant words and average path lengths of top words in bases. Table 5 shows Pearson correlations between average path length of top words and assignment evaluations (MAP, TAB). The moderate negative correlation implies that the quantities move in opposite directions (as expected).

Polysemous words occur in all sparse embeddings with their multiple meanings reflected by the assigned concepts. For example, *court* is a dominant word of bases that are assigned to meta-concepts *{law, legal}* and *{sport}*. Likewise, *virus* is dominant for bases assigned to meta-concepts *{computer, network, desktop}* and *{disease, pathology}*.

Altogether, there are 63 meta-concepts (for 119 separate concepts) assigned to some base in every embedding. Comparison of the three sparse embedding approaches with respect to concepts is given in Table 6. K-means tends to have bases where the words with the highest coefficients are actually associated with the assigned concept. This corresponds to the quantitative results (see Section 4.2). On the other hand, as seen in Table 4, GMPO has bases with dominant words that are not connected to the assigned concept of a given base, but there is a semantic relation between them (*tesla* is an automotive company, *juno* is the Roman equivalent of Hera, *retroactive* is a type of law). Also, Table 6 shows an example for DLSC where the assignment is wrong: *porgy, tchaikovsky, bluebeard, falstaff, ariadne* are rather connected to opera and not Greek mythology or Greek god.

Table 6: The 3 highest nonzero coefficient words for assigned concepts in the three sparse embeddings. The words that appear in ConceptNet alongside the assigned concept are **bold**.

concept(s)	DLSC	k-means	GMPO
car, cars	**sedan chevrolet bmw**	**sedan hatchback coupe**	tesla **roadster** musk
disease, pathology	**disease diseases encephalitis**	**measles polio diphtheria**	**polio measles** immunization
greek mythology, greek god	porgy tchaikovsky bluebeard	**zeus theseus** odin	juno award gemini
law, legal	**judge court appellate**	**appellate court** supreme	**waiver** retroactive infielder
mathematics	**polynomial** integer **invertible**	**abelian topological affine**	integer **factorization polynomial**

7 Conclusions and future work

In this paper we analyzed the extent to which the bases of sparse word embeddings overlap with commonsense knowledge. We provided an algorithm for labeling the most dominant semantic connotations that the individual bases convey relying on ConceptNet. Our qualitative experiments suggest that there is substantial semantic content captured by the bases of sparse embedding spaces. We also demonstrated the semantic coherence of the individual bases via analysing the paths between concepts in ConceptNet and quantified the correlation between the two types of evaluations.

Our experiments suggest several directions. Construction methods for sparse word embeddings combining the approaches studied, such as k-SVD, could be added for comparison. We are planning to expand our analysis to dense embeddings as well. Concept assignment could be extended to include other forms of composite concepts and bases. Spreading activation and network analysis methods going beyond path lengths could be used to determine semantic relatedness, taking into account the "heaviness" information obtained, edge labels, combination with random walks, neighborhood analysis and other techniques; for example Diochnos in [10] explores several properties of ConceptNet 4 with the tools of network analysis and some of these findings can potentially be associated with providing meaning to word embeddings using more recent versions of ConceptNet. Experiments are planned on extending current techniques for downstream NLP tasks and knowledge base analysis using the explicit information found in the word embeddings.

Acknowledgements

This research was in part supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund. This work was in part supported by the National Research, Development and Innovation Office of Hungary through the Artificial Intelligence National Excellence Program (grant no.: 2018-1.2.1-NKP-2018-00008).

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006.

[2] Mohammed Alsuhaibani, Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. Jointly learning word embeddings using a corpus and a knowledge base. *Plos One*, 13(3):1–26, 2018.

[3] Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. *CoRR*, abs/1308.6273, 2013.

[4] Vanda Balogh, Gábor Berend, Dimitrios I. Diochnos, Richárd Farkas, and György Turán. Interpretability of Hungarian embedding spaces using a knowledge base. In *XV. Conference on Hungarian Computational Linguistics (2019)*, pages 49–62. JATE Press, 2019.

[5] Gábor Berend. Sparse coding of neural word embeddings for multilingual sequence labeling. *Transactions of the Association for Computational Linguistics*, 5:247–261, 2017.

[6] Tanya Berger-Wolf, Dimitrios I. Diochnos, András London, András Pluhár, Robert H. Sloan, and György Turán. Commonsense knowledge bases and network analysis. In *11th International Symposium on Logical Formalizations of Commonsense Reasoning*, March 2013.

[7] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *27th Annual Conference on Neural Information Processing Systems 2013*, pages 2787–2795, 2013.

[8] Allan M. Collins and Elizabeth F. Loftus. A Spreading-Activation Theory of Semantic Processing. *Psychological review*, 82(6):407, 1975.

[9] Allan M. Collins and M. Ross Quillian. Retrieval Time from Semantic Memory. *Journal of Verbal Learning and Verbal Behavior*, 8(2):240–247, 1969.

[10] Dimitrios I. Diochnos. Commonsense Reasoning and Large Network Analysis: A Computational Study of ConceptNet 4. *CoRR*, abs/1304.5863, 2013.

[11] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, May–June 2015.

[12] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, July 2015.

[13] Yue Feng, Ebrahim Bagheri, Faezeh Ensan, and Jelena Jovanovic. The state of the art in semantic relatedness: a framework for comparison. *Knowledge Eng. Review*, 32:e10, 2017.

[14] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 397–406, 2014.

[15] Goran Glavaš and Ivan Vulić. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45, 2018.

[16] Brian Harrington. A Semantic Network Approach to Measuring Relatedness. In *COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume*, pages 356–364, 2010.

[17] Harold Hotelling. Relations Between Two Sets of Variates. *Biometrika*, 28(3/4):321–377, 1936.

[18] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, July 2015.

[19] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, 2009.

[20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[21] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 303–308, 1993.

[22] Brian Murphy, Partha Talukdar, and Tom Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950, December 2012.

[23] Farhad Nooralahzadeh, Cédric Lopez, Elena Cabrio, Fabien Gandon, and Frédérique Segond. Adapting Semantic Spreading Activation to Entity Linking in Text. In *International Conference on Applications of Natural Language to Information Systems*, pages 74–90. Springer, 2016.

[24] Dominique Osborne, Shashi Narayan, and Shay Cohen. Encoding prior knowledge with eigenword embeddings. *Transactions of the Association for Computational Linguistics*, 4:417–430, 2016.

[25] 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, October 2014.

[26] M. Ross Quillian. The Teachable Language Comprehender: A Simulation Program and Theory of Language. *Communications of the ACM*, 12(8):459–476, 1969.

[27] Gerard Salton and Chris Buckley. On the Use of Spreading Activation Methods in Automatic Information Retrieval. In *SIGIR'88, Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval 1988*, pages 147–160, 1988.

[28] Lutfi Kerem Senel, Ihsan Utlu, Veysel Yücesoy, Aykut Koç, and Tolga Çukur. Semantic structure and interpretability of word embeddings. *CoRR*, abs/1711.00331, 2017.

[29] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Language Resources Association (ELRA), 2012.

[30] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. SPINE: sparse interpretable neural embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 4921–4928, 2018.

[31] Yulia Tsvetkov, Manaal Faruqui, and Chris Dyer. Correlation-based intrinsic evaluation of word vector representations. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 111–115, August 2016.

[32] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, September 2015.

Appendix

A Hardware used for the experiments

Experiments related to Sections 3, 4 and 5 were conducted on three different hardware environments that we detail next:

- Section 3: Intel(R) Xeon(R) CPU E7- 4820 @ 2.00GHz; 512Gb RAM (using $< 1\%$ of it)
- Section 4: Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz; 8GB RAM
- Section 5: Intel Core i5 CPU 5287U @ 2.9 GHz; 16GB RAM

B Pseudocode for GMPO algorithm

Algorithm 1: GREEDYMAXIMIZINGPAIRWISEORTHOGONALITY (GMPO)

Input: Embedding matrix $X \in \mathbb{R}^{|V| \times m}$ with word vectors ordered according to the frequency of the words they describe, k for the desired size of the dictionary matrix to return
Output: dictionary matrix D

$X \leftarrow$ UNITNORMALIZEROWS(X);
$D \leftarrow [\mathbf{x_1}]$; // $\mathbf{x_1}$ is the first row of the embedding matrix X
for $(i = 1; i < k; ++i)$ **do**
  $j^* \leftarrow \underset{j \in \{1,\ldots,|V|\}}{\arg\min} \max_{1 \le l \le i} |\langle \mathbf{x_j}, \mathbf{d_l} \rangle|$; // $|\cdot|$ is for taking elementwise absolute value
  $D \leftarrow [D;\ \mathbf{x_{j^*}}]$; // expand D with $\mathbf{x_{j^*}}$
return D;

C Overview of the base assignment algorithm

Algorithm 2 provides the base assignment algorithm in pseudocode.

Algorithm 2: BASE ASSIGNMENT (BA)

Input: sparse embedding matrix α, knowledge base kb
Output: $assignments$ containing the associated concepts to each base

$nodes \leftarrow \{(start, end)$ in $kb\}$;
$C \leftarrow$ `BIADJACENCY(`*nodes*`)`;
$A \leftarrow$ `TRANSPOSE(`C`)` $*$ `BINARIZE(`α`)`;
$P \leftarrow$ `NPPMI(`A`)`;
$maxConcepts \leftarrow$ `ARGMAX(`*P, maxBy=columns*`)`;
$assignments \leftarrow$ `CLOSECONCEPTS(`*maxConcepts,* `NPPMI(TRANSPOSE(`C`)*`C`))`;
return $assignments$;

D Omitted discussion from Section 5

Below we provide additional information to the discussion that we have in Section 5.

D.1 Omitted discussion on average path lengths

Here we give some examples that explain the summary that we provided in the paragraph about average path lengths in Section 5. All four examples below come from the GMPO dictionary.

- In base 2, the top 10 words are related in one way or another to *anatomy*: distal, proximal, apical, dorsal, posterior, ventral, anterior, tubule, humerus, basal. Indeed, the average path

length between them is 2.222 and the critical concept for connecting these words, if they are not directly connected, is the word *anatomy*. Furthermore, we may find some paths that connect the given words by using a different meaning for some of them. For example, *apical* gives the path /c/en/apical – /c/en/phonetics – /c/en/front – /c/en/anterior. Hence, even if polysemy can be an issue, in this case a fairly short path (of length 3) was found.

- In base 582, the average path length of the top 10 words associated with this base is 2.6 and these top words are: riesling, cabernet, alsatian, alsace, syrah, chardonnay, fruity, viognier, zinfandel, merlot. In this case, fruity is not a wine variety but rather may characterize wine of a certain variety.
- In base 9, the top words are: karakoram, pamir, hindu, mountains, sunda, bora, "diablo", andes, nubia, himalayan. These words are mostly about mountains and yield an average path length of 3.022.
- In base 19, the top words are: mvp, honorable, acc, selection, preseason, varsity, sophomore, freshman, earning, consulship. These words are certainly not as coherent as in the previous two examples. Nevertheless, the average pairwise distance among them is 3.0 and in fact this is shorter compared to the average path length among the 10 bottom words for the particular base (3.578).

D.2 Description of the algorithms used

Algorithms 3 and 4 describe respectively the processes of (i) spreading activation along the network, and (ii) computing a heavy path connecting two vertices s and t in the activated network given a meeting point along that path.

Regarding spreading activation we use the following default values:

Default initial weight (activation): The value is 0.1. This is given to the two nodes s and t for which we want to compute a path.

Decay factor: The value is 0.01.

Firing threshold: 0.00001.

Refire constant: 1.

Max spreading activeation iters: 20.

D.3 A normal form for the words in ConceptNet.

We create some equivalence classes for text that corresponds to certain words. The words that we obtain from the dictionary methods are in English language (e.g., *dog*). However, for ConceptNet we map the word *dog* to the word */c/en/dog*. Furthermore, when we want to obtain the neighbors of such a word in ConceptNet (as in line 6 of Algorithm 3) we look for neighbors for the words that are obtained by appending the suffixes "/a", "/n", "/r", "/v". So, in particular, if we want to obtain the neighbors (incoming and outgoing edges) of the word *dog* we query ConceptNet for all five variants: /c/en/dog, /c/en/dog/a, /c/en/dog/n, /c/en/dog/r, /c/en/dog/v.

Note that we still apply these suffixes for words that belong to other languages as well. For example in the end of Section 5 we have the French word /c/fr/géologie and as a consequence, in such a situation we are looking for the neighbors of all five cases: /c/fr/géologie, /c/fr/géologie/a, /c/fr/géologie/n, /c/fr/géologie/r, /c/fr/géologie/v.

D.4 More on the experimental setup

ConceptNet 5.6 was obtained from

https://github.com/commonsense/conceptnet5/wiki/Downloads

and it was installed on MongoDB v4.0.3. ConceptNet 5.6 has 32,755,210 assertions. A typical record (assertion) is stored as a JSON object in the format shown below.

Algorithm 3: Spread Activation Along the Network

Input: A network G and two ('raised') vertices s and t for which we want to compute a path in G
Output: A subgraph of G containing 'raised' vertices; among these nodes a path between s and t will be found by using nodes from this graph.

foreach *raised vertex* v **do**
  $v.tempWeight \leftarrow v.spreadingWeight$;
  $v.tempLabels \leftarrow v.labels$;
foreach *raised vertex* v **do**
  if $(v.requestFireNode\ is\ TRUE)\ AND\ (v.timesFired < max\_refire\_constant)$ **then**
    foreach *neighbor* u *of* v *(or* v*'s variants);* `// See Section D.3 for the 'variants'` **do**
      Add u in the raised graph;
      $u.tempWeight \leftarrow u.tempWeight + v.spreadingWeight \cdot decay\_factor$;
      if $u.tempWeight \geq firing\_threshold$ **then** $u.requestFireNodeNextRound \leftarrow TRUE$;
      Append $v.labels$ to $u.tempLabels$;
      Update $u.distance\_from\_s$ based on $v.distance\_from\_s$;
      Update $u.distance\_from\_t$ based on $v.distance\_from\_t$;
    $v.timesFired \leftarrow v.timesFired + 1$;
$met \leftarrow FALSE$;
foreach *raised vertex* v **do**
  $v.spreadingWeight \leftarrow v.tempWeight$;
  $v.labels \leftarrow v.tempLabels$;
  if $|v.labels|$ *is* 2 **then** $met \leftarrow TRUE$;
  if $v.requestFireNodeNextRound\ is\ TRUE$ **then** $v.requestFireNode \leftarrow TRUE$;
  else $v.requestFireNode \leftarrow FALSE$;
  $v.requestFireNodeNextRound \leftarrow FALSE$;
if $met\ is\ TRUE$ **then**
  There is at least one path between s and t, so stop with success here
else if $current\_iteration > max\_spreading\_activation\_iters$ **then**
  Stop here with failure as we did not manage to find a path between s and t
else
  Apply one more round of spreading activation


```
{
    "_id" : ObjectId("5bcd6acb1030aeeb19c8967c"),
    "dataset" : "/d/conceptnet/4/en",
    "license" : "cc:by/4.0",
    "sources" : [
        {
            "activity" : "/s/activity/omcs/commons_manual_entry",
            "contributor" : "/s/contributor/omcs/sandos"
        }
    ],
    "surfaceEnd" : "quadriped",
    "surfaceStart" : "dog",
    "surfaceText" : "[[dog]] is a kind of [[quadriped]].",
    "weight" : 1,
    "uri" : "/a/[/r/IsA/,/c/en/dog/,/c/en/quadriped/]",
    "rel" : "/r/IsA",
    "start" : "/c/en/dog",
    "end" : "/c/en/quadriped",
    "id" : 16412260
}
```


We have also created indices in the collection for the properties 'start' and 'end' of the JSON documents. This is done so that we can obtain quickly the neighbors of the various words that we want (during spreading activation, or otherwise).

Algorithm 4: Find Heavy Path

Input: The raised subgraph of G that Algorithm 3 generated, vertices s and t, a meeting vertex m
Output: An undirected path connecting s to t going through m

$current\_node \leftarrow m$;
$p_s \leftarrow (m)$;
while $current\_node \neq s$ **do**
    Get predecessors $Pred$ of $current\_node$;
    $next\_node \leftarrow u \in Pred$ such that $u.spreadingWeight$ is max among those vertices in $Pred$;
    Augment p_s with $next\_node$;
    $current\_node \leftarrow next\_node$;
$current\_node \leftarrow m$;
$p_t \leftarrow (m)$;
while $current\_node \neq t$ **do**
    Get predecessors $Pred$ of $current\_node$;
    $next\_node \leftarrow u \in Pred$ such that $u.spreadingWeight$ is max among those vertices in $Pred$;
    Augment p_t with $next\_node$;
    $current\_node \leftarrow next\_node$;
Concatenate p_s and p_t to form a heavy path p connecting s and t and return that path p;

For the scripts that we use in order to query the database and perform computations (as well as implement Algorithms 3 and 4), we use Node.js; in particular version 8.12.0. The script that we run in order to apply spreading activation for the various starting points and subsequently compute heavy short paths connecting the starting points is allowed to use up to 14GB of RAM and we also increase the allowed size to be used for the stack during the execution. In particular, we perform our results with the command shown below.

```
node --max_old_space_size=14000 --stack-size=14000 query_conceptnet56.js
```

Other dependencies that we use are the following (found in the package.json file).

```
"dependencies": {
    "mongodb": "3.1.8",
    "async": "2.6.1",
    "assert-plus": "1.0.0"
}
```