Word Equations: Inherently Interpretable Sparse Word Embeddings through Sparse Coding

Adly Templeton
Williams College
adlytempleton@gmail.com

Abstract

Word embeddings are a powerful natural language processing technique, but they are extremely difficult to interpret. To enable interpretable NLP models, we create vectors where each dimension is *inherently interpretable*. By inherently interpretable, we mean a system where each dimension is associated with some human-understandable *hint* that can describe the meaning of that dimension. In order to create more interpretable word embeddings, we transform pretrained dense word embeddings into sparse embeddings. These new embeddings are inherently interpretable: each of their dimensions is created from and represents a natural language word or specific grammatical concept. We construct these embeddings through sparse coding, where each vector in the basis set is itself a word embedding. Therefore, each dimension of our sparse vectors corresponds to a natural language word. We also show that models trained using these sparse embeddings can achieve good performance and are more interpretable in practice, including through human evaluations.

1 Introduction

Word embeddings represent each word in a natural language as a vector in a continuous high dimensional space. Many different pretrained embeddings are readily available (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017), and are used in a range of applications (Li and Yang, 2018). This vector representation can be said to encode the meaning of the word; not only are similar words close together but linear relationships between words are thought to have conceptual meaning. In the famous example, the vector difference between 'man' and 'woman' is similar to the vector difference between 'king' and 'queen' (Landauer and Dumais, 1997; Mikolov et al., 2013). This observation suggests that the vector difference between 'woman' and 'man' represents a concept of gender within the vector space, implying that dimensions or linear combinations of dimensions in the vector space are related to human-understandable concepts. However, in practice, interpreting these vector spaces is extremely difficult. This obscures the behavior of any NLP model built on top of word embeddings.

To enable interpretable NLP models, we create vectors where each dimension is *inherently interpretable*. By inherently interpretable, we mean a system where each dimension is associated with some human-understandable *hint* that can describe the meaning of that dimension. This allows us to directly interpret the coefficients of simple models trained on these vectors. By comparison, most other systems of interpretable word embeddings aim to create dimensions that humans may be able to manually interpret after the fact.

To create our vectors, we represent word embeddings as the sparse linear combination of a basis set of other word embeddings. Our primary contribution is that, instead of learning an optimal basis for our sparse vector space, we draw the columns of the basis from the original set of dense word embeddings. This strategy provides a natural label for each sparse dimension and allows us to represent each natural language word as the linear combination of a small number of other natural language words. This representation is itself a more 'interpretable' word embedding. This technique produces representations of words that have interpretable dimensions. We show that these representations are more interpretable and that models trained on these embeddings perform almost as well as models trained on standard dense embeddings. We show how the creation of inherently interpretable vectors can help us understand the behavior and structure of the original word embeddings.

Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 177–191
Online, November 11, 2021.

Recent work has created more interpretable vectors through a variety of methods. However, relatively few approaches create *inherently interpretable* dimensions. Therefore, we believe that our work, which creates inherently interpretable embeddings through a simple novel method can be the basis of future NLP tools where interpretability is crucial.

As an example, we present one randomly selected embedding from our system. More examples can be found in the appendix.

$$\begin{aligned}
&\text{carbon} = 0.79 * \text{nitrogen} \\
&- 0.38 * \text{CAPITALIZATION} + 0.3 * \text{fossil} \\
&- 0.21 * \text{POS-NOUN} + 0.16 * \text{POS-ADJ} \\
&+ 0.14 * \text{C0} - 0.14 * \text{PAST-TENSE} \\
&+ 0.13 * \text{wood} + 0.11 * \text{global} \\
&+ 0.1 * \text{atoms} - 0.095 * \text{POS-ADV} \\
&+ 0.092 * \text{aluminum} \\
&- 0.078 * \text{PLURAL-NOUN} \\
&+ 0.073 * \text{greenhouse} \\
&- 0.072 * \text{POS-PROPN} \\
&- 0.048 * \text{POS-VERB} + 0.046 * \text{forestry} \\
&+ 0.03 * \text{PARTICIPLE} + 0.017 * \text{sink} \\
&+ 0.012 * \text{POS-NUM}
\end{aligned}$$

2 Previous Work

Park et al. (Park et al., 2017) find a more interpretable rotation of word embeddings using techniques associated with factor analysis. Other work (Dufter and Schütze, 2019; Rothe and Schütze, 2016) rotates dense vectors using different methods.

Koc et al. (Şenel et al., 2020) tie concepts to dimensions in a more direct way. They select a concept for each dense dimension and identify words that are associated with these concepts. A penalty term pushes coefficients for these words towards the fixed values.

Other work has focused on interpretability through sparsity. Subramian et al. (Subramanian et al., 2018) created more interpretable embeddings by passing pretrained dense embeddings through a sparse autoencoder.

Panigrahi et al. (Panigrahi et al., 2019) proposed Word2Sense, a generative approach that models each dimension as a 'sense' and word embeddings as a sparse probability distribution over the senses.

The mathematical technique we use in this paper, *Sparse coding*, which is defined as the representation of vectors as the sparse linear combination of an overcomplete basis, is a well-studied optimization problem (Coates and Ng, 2011; Hoyer, 2002; Lee et al., 2007). Previous work (Coates and Ng, 2011) has also shown that basis vectors can be efficiently selected from the set that is being encoded.

Faruqui et al. (Faruqui et al., 2015) used non-negative sparse coding to recode dense word embeddings into more interpretable sparse vectors while learning a basis. However, because they create their basis through direct optimization, the basis vectors (and, consequently, the dimensions in their transformed sparse space) do not have any inherent interpretation and must be manually interpreted.

Zhang et al. (Zhang et al., 2019) also used non-negative sparse coding to learn a set of *word factors* to recode word2vec embeddings. The basis vectors created in this way are highly redundant, so they then use spectral clustering to remove near-duplicate factors. Then, they are able to manually infer reasonable post hoc interpretations for most of the factors.

Concurrently with our work, Mathew et al. create an inherently interpretable subspace from pairs of antonyms. They then project embeddings into that subspace, producing lower-dimensional dense vectors (Mathew et al., 2020).

3 Model

Our work uses *sparse coding* to transform a set of word embeddings from a dense and uninterpretable space into a sparse and interpretable space. Let v_D represent a dense word embedding, and let $\mathcal{B}$ represent a matrix with basis vectors along the columns. $\mathcal{B}$ has size (n_S, n_N) where n_d is the dimensionality of the dense vectors and n_S is the dimensionality of the sparse vectors. We achieve sparse coding using regularized regression, inducing sparsity using the L_1 norm. Formally, this corresponds to finding the sparse vector v_S that minimizes the following objective function

$$\arg\min_{v_S} \; ||v_D - v_S\mathcal{B}||_2^2 + \alpha||v_S||_1 \qquad (1)$$

α is a hyperparameter that controls the level of sparsity. The first term in Equation 1 ensures the sparse vector corresponds to a vector in the dense space that is similar to the original vector. The second term is a sparsity-inducing penalty.

Note that by 'basis' we mean a set of vectors in the dense space, each one corresponding to a dimension in the transformed, sparse, space. Out of necessity, these vectors are *overcomplete* (there are more dimensions than vectors) and so they do not form a basis according to the traditional definition.

Previous work using sparse coding to create interpretable word embeddings has considered the basis $\mathcal{B}$ to be part of the optimization problem (Faruqui et al., 2015; Zhang et al., 2019). Our primary contribution is that, instead of learning an optimal basis, we draw the columns of the basis from the original set of dense word embeddings. This strategy provides a natural label for each sparse dimension.

3.1 Grammatical Basis

We can roughly divide the 'meaning' carried by a word embedding into *grammatical* and *non-grammatical* properties. Here we use 'grammatical properties' to mean properties that describe how that word fits into the grammar of the language, such as its part-of-speech, tense, or number. We use 'non-grammatical properties' to mean all other aspects of the meaning of a word. For instance, we expect the embedding for the word 'swimming' to include a grammatical component representing that this word is a present-tense participle and a non-grammatical component that represents the meaning 'to swim'. Of course, this deconstruction is imperfect. Nevertheless, this approach provides a useful insight towards decomposing the meaning of a word embedding.

Preliminary experiments showed that, without special consideration, grammatical properties would be captured in an unintuitive way. The grammatical components could not be easily isolated to one subset of the nonzero dimensions. Ideally, the grammatical information would be captured in a small number of interpretable dimensions. Instead, each basis vector would capture part of the grammatical component and part of the semantic component. This duality creates difficulty when interpreting our representations.

To address this, we construct a small number of *grammatical basis vectors* and add them to the basis set. For instance, we construct a 'POS-NOUN' vector by taking the mean of all word embeddings corresponding to nouns. For this work, we use a set of 11 grammatical basis vectors, though the number and the construction of these are arbitrary. A description of the grammatical basis vectors is in the appendix.

Next, we make the grammatical basis vectors orthogonal using the Gram-Schmidt process. Finally, we subtract the projection along the grammatical basis vectors from all other ('non-grammatical') basis vectors we use and renormalize them. This procedure separates the grammatical meaning from our non-grammatical basis vectors, ensuring that non-grammatical bases are not also coding for grammatical concepts.

Note that we only perform this orthogonalization with respect to a very small number of grammatical basis vectors. We find that this procedure does not remove more than 50% of the length of any individual vector and 50% of vectors have less than 20% of their length removed.

When encoding a dense vector, instead of finding the grammatical coefficients using sparse coding, we set each grammatical coefficient to the projection along the corresponding grammatical basis vector, which is equal to the dot product similarity between the original vector and the grammatical basis vector. Because the grammatical basis is orthogonal, we can do this for every grammatical basis vector simultaneously. This residual is then transformed using Equation 1.

Note that, although we do require hand-crafted features to create the grammatical basis vectors, our system does not use hand-crafted features in the representation of new words. Once the grammatical feature vectors are defined, words can be represented in our sparse space using no more information than their *fasttext* dense vectors.

3.2 Basis Selection

We cannot practically use all words as our basis set, so we have to select a subset. First, we start with the 30,000 most frequent words. We filter out any words that are capitalized or that are not in a standard English vocabulary (using the vocabulary of the spaCy *en_core_web_sm* model). Next, we filter out any words that are not nouns, verbs, or adjectives. This process removes many basis vectors that may be hard to interpret. This gives us approximately 11,000 remaining potential basis words. From these, we will select 3,000 words to use in the final basis.

We use an iterative algorithm that takes, at each step, the potential basis vector with the highest mean cosine similarity to all other vectors. To

encourage diversity, this mean is weighted by the lowest cosine dissimilarity that each vector has with any already-selected basis vector. Formally, at each step, we grow the set of basis vectors B by adding the potential basis vector x from the set of unchosen potential basis vectors $\mathcal{F} \setminus B$ that satisfies

$$\arg\max_{x \in \mathcal{F} \setminus B} \sum_{v \in V_D} (x \cdot v) \max_{b \in B} (1 - b \cdot v)$$

Where V_D is the set of dense vectors for the 30,000 most frequent words.

Note that, despite our use of the word 'basis', this is not a basis in the traditional sense; the set of basis vectors are not linearly independent, and there are more basis vectors than dimensions in the original space. However, because of the L1 penalty term, our objective function still allows for optimal decompositions.

3.3 End-to-end Process

In order to find the sparse vector representation, we follow the following process, combining the above elements.

1. Find the dense vector representation of the word.

2. Compute the projection along each vector orthogonal grammatical basis. Store these projections as the first part of the resulting vector. Subtract the projection along this basis before moving on to the next step.

3. Optimize Eq. 1 using the FISTA algorithm (Chalasani et al., 2013). Store the learned sparse vector as the second part of the resulting vector.

4 Results

We will evaluate this model in multiple ways. In particular, we care about two contradictory properties of our transformed vector space. First, we want our vector space to be useful in downstream machine learning applications. We expect that, in most applications, increased interpretability comes with some performance cost. Therefore, we care about the performance loss when moving from dense vectors to our sparse vectors.

The other goal is that our sparse vectors should be interpretable. It is much harder to articulate exactly what interpretability is or how we can measure it. Metrics such as the Word Intrusion Task (Section2) can act as a useful proxy for interpretability, and we use it as our primary quantitative measure of interpretability. But part of interpretability is, by definition, subjective and any metric is imperfect.

4.1 implementation

We use the FastText (Bojanowski et al., 2017) pretrained 300 dimensional English vectors (without subword information) trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset as the dense vectors that we input into our models. Unless otherwise mentioned, we only consider the 30,000 most frequent words, for computational reasons. We normalize all vectors to have mean 0 and unit length. After learning sparse vectors, we normalize each sparse vector so that it corresponds to a dense vector of unit length. When comparing with the original dense vectors (FastText (Bojanowski et al., 2017)), we subtract the mean of all vectors, to match our preprocessing.

In practice, the sparse penalty term will only push coefficients very close to 0. We clamp any coefficient with a magnitude of less than .001 to 0. We found this threshold by taking the lowest cutoff that does not introduce significant irregularities into the tradeoff curves in Section 4.3.

We solve the regularized optimization problems using the FISTA algorithm (Chalasani et al., 2013), as implemented in the Python Lightning package (Blondel and Pedregosa, 2016), using default hyperparameters. FISTA is an optimization algorithm that can efficiently solve sparse coding problems. We use the spaCy library (Honnibal and Montani, 2017) to check for out of vocabulary words and perform part-of-speech tagging. We use the numpy (Oliphant, 2006), CuPy (Okuta et al., 2017), and Scikit learn (Pedregosa et al., 2011) libraries for various linear algebra implementations. We use the open-source Gensim library (Rehurek and Sojka, 2010) to manipulate word embeddings. For the word analogy task evaluation, we use the *3CosAdd* method, as implemented by Gensim. Models processed 30,000 words within a few hours, running across 32 2.5 GHz processors with no GPU.

4.2 Comparison with Previous Work

To compare our work against other sparse coding approaches, we will often reference the vectors created by Faruqui et al. (Faruqui et al., 2015). That work generates more interpretable vectors using

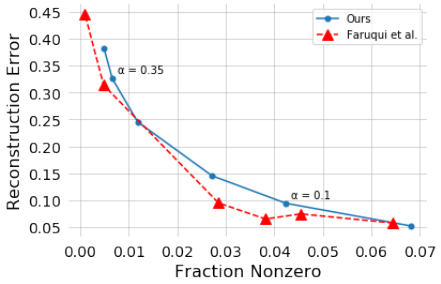

Figure 1: The tradeoff curve between sparsity and reconstruction error. The dashed line shows the tradeoff curve achieved by Faruqui et al. using sparse coding without inherently interpretable dimensions (Section 4.2).

sparse coding but without inherently interpretable dimensions.

4.3 Reconstruction Error and Sparsity

Note that, because of the penalty term in Equation 1, $V_S\mathcal{B}$ (the *reconstructed vectors*) are not exactly equal to the original dense vectors V_D. Therefore, we expect a tradeoff between sparsity and this difference (which we call *reconstruction error)*.

This tradeoff curve is displayed in Figure 1. Despite the additional constraints of an inherently interpretable system, we suffer only a minor increase in reconstruction error compared to traditional sparse coding. This *reconstruction error* is the primary drawback of our system; reconstruction error adds a small amount of noise to every model built on top of our sparse vectors. For the remainder of this work, unless otherwise mentioned, we will consider the vectors made with $\alpha = 0.35$. These vectors have, on average, 20 nonzero entries.

4.4 Analogy Task

In the word2vec vector space, famously, the vector for 'king' plus the vector for 'woman' minus the vector for 'man' is close to the vector for 'queen'. Analogy tasks quantitatively test these properties. The task consists of analogies of the form *A is to A′ as B is to B′*. The vector space is evaluated on its ability to correctly determine the value of B'.

The performance of our vector space at this task is displayed in Table 1. Our model performs poorly on this task. This degradation comes from two sources. First, the drop from the original vectors to the reconstructed vectors that is due to reconstruction error. Second, an additional degradation is caused by the transformation from dense vectors to sparse vectors, especially with cosine similarity.

	Nonzero	Total	Gram.	Sem.
FastText	300	0.88	0.85	0.94
Faruqui $\lambda = 0.75$	136	0.65	0.60	0.73
Ours $\alpha = 0.1$	127	0.50	0.46	0.59
Recons. $\alpha = 0.1$	300	0.83	0.80	0.88
Ours $\alpha = 0.35$	20	0.20	0.22	0.15
Recons. $\alpha = 0.35$	300	0.33	0.38	0.25

Table 1: Accuracy on the word2vec analogy evaluation set for various vector spaces. The first column shows the average number of nonzero entries in each sparse vector. Accuracy is also broken down by non-grammatical and grammatical categories. 'Recons' denotes the performance of the reconstructed dense vectors. All results are on a 50% held-out test set.

4.5 Classification

Next, we demonstrate that our model can be used to build interpretable machine learning systems. To this end, we train classifiers using our word embeddings as input. We demonstrate that these classifiers are not only effective but also interpretable.

We evaluate our vectors on two datasets, the IMDB sentiment analysis dataset (Maas et al., 2011) and the TREC question classification dataset (Li and Roth, 2002). For both of these datasets, we use a logistic regression model and a bag of words representation.

4.5.1 IMDB Sentiment Analysis Dataset

The IMDB movie review dataset consists of 50,000 passages taken from IMDB movie reviews, evenly split between positive and negative reviews. The task is to determine the sentiment of each passage (Maas et al., 2011).

We train classifiers using various word embedding spaces as inputs. While we could train deep neural modesl on these vector spaces, neural models do not directly produce interpretable coefficients, and therefore we provide a demonstration on simple logistic regression models. The results are presented in Table 2. Our vector spaces demonstrate improvement over the original dense vectors (FastText (Bojanowski et al., 2017)), as well as the traditional sparse coding approach of Faruqui et al. This result holds despite a slight decrease in performance caused by the reconstruction error (as demonstrated by the low performance with reconstructed vectors).

We can directly interpret our classifier's coefficients. Here, we present the most significant coeffi-

	IMDB	TREC
FastText	85.35	84.2
Faruqui $\lambda = .75$	85.54	84.4
Ours $\alpha = 0.1$	**87.51**	**86.2**
Ours Recons. $\alpha = 0.1$	85.08	81.4
Ours $\alpha = 0.35$	86.46	84.0
Ours Recons. $\alpha = 0.35$	83.00	75.8

Table 2: Accuracy on the IMDB sentiment analysis dataset and the TREC question classification dataset. We use a logistic regression classifier, which uses as input a bag-of-words sum of various word embeddings.

cients ($\alpha = 0.1$) [1]:

$$\ln\frac{P(\text{positive})}{1-P(\text{positive})} = -157 \cdot \text{dreadful} - 153 \cdot \text{horrible} + 150 \cdot \text{fabulous} - 140 \cdot \text{dull} - 132 \cdot \text{dreary} - 107 \cdot \text{worsen} - 105 \cdot \text{ridiculous} + ...$$

Note that these are not coefficients on the frequencies of individual words. Instead, these are coefficients on vectors in the basis set. We can consider them to be coefficients on concepts, which are labeled by the displayed words. The coefficients make sense: positive concepts have positive coefficients, while negative concepts have negative coefficients. This pattern continues for much longer than displayed above, and we have omitted other terms for space reasons. The first term to not fall into this clear interpretation is the 24th-most significant: $... + 74 \cdot \text{shall} + ..$‘

At first, this term appears nonsensical. Looking more closely at this dimension can reveal more about our system. The top five words in the dimension represented by ‘shall’ are the following: ‘henceforth’, ‘herein’, ‘hereafter’, ‘thereof’, ‘hereby’. We can see here how both our vector space and our regression model pick up on tone. This dimension appears to correspond to a formal and somewhat archaic tone, which is likely not found in a negative internet comment.

4.5.2 TREC Question Classification Dataset

Our next classification task is more complex. The TREC question classification dataset consists of 6,000 questions that are divided into 6 categories based on the expected answer: abbreviations, descriptions, entities, humans, locations, and numeric.

Accuracy for various vector spaces is presented in Table 2. Again, our model does better than the unmodified input vectors we start with, despite some loss from the reconstruction error. Both results suggest that our vector spaces are efficient in regression-based settings, though the performance at the word-analogy task suffers a serious degradation. It is likely that different qualities are needed for these different tasks. The exact-match evaluation of the word analogy task severely punishes even slight noise in the vector space, and cosine similarities are noisy in sparse vectors.

Once again, we directly interpret the coefficients learned by logistic regression. For space, we display the most significant terms for the HUM category. Questions in this category expect the name of a human as the answer:

$$\ln\frac{P(\text{HUM})}{1-P(\text{HUM})} = -77 \cdot \text{wonder} + 66 \cdot \text{organizations} + 54 \cdot \text{companies} + 51 \cdot \text{poet} + 49 \cdot \text{songwriter} + 48 \cdot \text{identities} + 42 \cdot \text{fan} - 42 \cdot \text{movie} - 39 \cdot \text{resulting} + 36 \cdot \text{university} - 36 \cdot \text{diseases} + 36 \cdot \text{successive} + 35 \cdot \text{consist} + 35 \cdot \text{cabinets} + ...$$

Some of these coefficients, such as ‘songwriter’ or ‘identities’ are intuitive and reveal interesting behavior of the classifier. Others, such as ‘wonder’, are not. Manual inspection reveals that ‘wonder’ is used to represent words such as ‘How’ or ‘why’ but not ‘who’, though this behavior is likely noise.

4.5.3 Word Intrusion Task

To quantitatively measure interpretability, we use human experiments. In particular, we use the word intrusion task (Chang et al., 2009). In this task, humans are presented with five words, four of which are associated highly with a particular dimension. Participants are asked to choose the word that does not belong. We use our vectors both with and without providing the label of the dimension as a ‘hint’.

We use the following procedure for generating questions. First, we filter candidate words, starting with the 20,000 most frequent words and filtering out words that are not lowercase, words that are

[1] These weights are real-values and truncated for space. Note that the weights are very large because they correspond to sparse low-magnitude features.

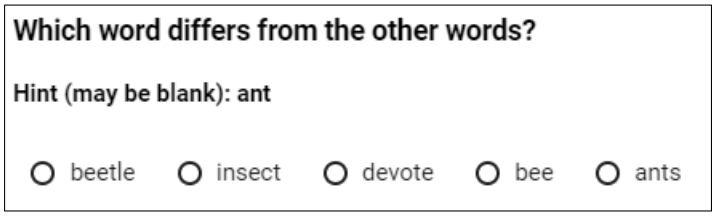

Figure 2: An example of the user interface given to annotators. The following instructions were given to the annotators: 'You will be presented with a group of 5 words. Four of these words are similar in some way and the other one is not. Pick out the word which is dissimilar. You may be provided with a hint about how the words are similar.'

	Accuracy	CI
FastText	0.31	[0.27,0.34]
Faruqui	0.77	[0.74,0.80]
Ours	0.80	[0.77,0.83]
Ours (with Hints)	0.84	[0.81,0.86]

Table 3: Results on the word intrusion task. 95% normal confidence intervals are displayed.

not made up of only ASCII alphabetic characters, and words with only one letter. Then we randomly select a dimension. We pick the 4 highest words along that dimension, and one word randomly selected from the bottom 50% of words in that dimension, then randomize the order. Each example is presented to three different Mechanical Turk annotators. An example of the interface presented to annotators is seen in Figure 2.

The results of the word intrusion task are presented in Table 3. When hints are provided, we see a statistically significant improvement in accuracy between our vectors and the sparse coding baseline ($p = .00055$). In addition, using hints produces a statistically significant improvement ($p = .040$), validating our motivation for inherently interpretable dimensions. Of course, any quantitative metric of interpretability is imperfect. To qualitatively assess interpretability, randomly selected vectors are presented in the appendix.

4.6 Summary

Our method still has some serious drawbacks. Sparse coding, by its nature, introduces a substantial amount of noise in the form of reconstruction error and sparse coding has the potential to assign very different sparse vectors to similar dense vectors. We hope that future work will produce sparse embeddings that are interpretable by construction without some of the shortcomings of our work.

4.7 Conclusions and Future Work

In this work, we presented a method to create word embeddings that are interpretable by construction. Each dimension of these embeddings corresponds precisely to a natural language word. These embeddings can be presented in a human readable form, and we have shown that most of these representations are intuitive. We have also shown that these embeddings can be used to produce an extremely interpretable classification model that still delivers performance comparable to or better than a classification model based on the original embeddings.

Unlike most previous work on interpretable word embeddings, our method does not require humans to interpret and label each dimension. We have previously seen how this feature allows us to easily create interpretable classification models. It also allows us to gain a deeper understanding of the original dense vector space. Previous approaches may have obscured nuanced or hard to interpret behavior. In particular, a human manually interpreting a dimension may not appreciate subtle behavior of the system. Several sections of this work, which have manually examined individual word representations in our system, have revealed the nuanced behavior that our system demonstrates.

Our method still has some serious drawbacks. While we have examined a number of these flaws, many are tied closely to the sparse coding method we have chosen to use. Sparse coding, by its nature, introduces a substantial amount of noise in the form of reconstruction error. In addition to the reconstruction error, sparse coding has the potential to assign very different sparse vectors to similar dense vectors. We hope that future work will produce sparse embeddings that are interpretable by construction without some of the shortcomings of our work.

Much of the promise of sparse coding methods remains to be proved. In particular, we believe it

will be fruitful to study the representation of syntactic concepts. We have seen that our attempts to disentangle syntactic concepts from our semantic basis vectors were not entirely successful. We would also like to better understand how these methods are applicable in deep learning models.

There is still a large amount of analytical work left to be done on evaluation. The word intrusion task, while an effective quantitative method, does not offer a complete view of interpretability. Part of this problem is that we do not have any way to quantify interpretability where it is most useful: when building downstream classification models. More fundamentally, we do not have any underlying framework for understanding what it means for a word embedding to be interpretable.

We believe that interpretable word embeddings have great potential for helping us understand and interpret models in a wide range of NLP tasks.

Acknowledgements

Thanks to Duane Bailey for his extensive support and advice, and for advising the thesis on which this paper is based.. Thanks to Andrea Danyluk for her guidance as the second reader of that thesis.

References

Mathieu Blondel and Fabian Pedregosa. 2016. Lightning: large-scale linear classification, regression and ranking in Python.

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

Rakesh Chalasani, Jose C Principe, and Naveen Ramakrishnan. 2013. A fast proximal method for convolutional sparse coding. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5. IEEE.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296.

Adam Coates and Andrew Y. Ng. 2011. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928.

Philipp Dufter and Hinrich Schütze. 2019. Analytical methods for interpretable ultradense word embeddings. *arXiv preprint arXiv:1904.08654*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. 2015. 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.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Patrik O. Hoyer. 2002. Non-negative sparse coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565. IEEE.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. 2007. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Yang Li and Tao Yang. 2018. Word embedding for understanding natural language: a survey. In *Guide to Big Data Applications*, pages 83–104. Springer.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Binny Mathew, Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. 2020. The polar framework: Polar opposites enable interpretability of pre-trained word embeddings. *arXiv preprint arXiv:2001.09876*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. 2017. CuPy: A NumPy-compatible library for nvidia gpu calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.

Travis E. Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. 2019. Word2sense: Sparse interpretable word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5692–5705.

Sungjoon Park, JinYeong Bak, and Alice Oh. 2017. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–517.

Lütfi Kerem Şenel, Ihsan Utlu, Furkan Şahinuç, Haldun M Ozaktas, and Aykut Koç. 2020. Imparting interpretability to word embeddings while preserving semantic structure. *Natural Language Engineering*, pages 1–26.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Juexiao Zhang, Yubei Chen, Brian Cheung, and Bruno A. Olshausen. 2019. Word embedding visualization via dictionary learning. *arXiv preprint arXiv:1910.03833*.

5 Appendix

A Gramtacic Basis Descriptions

Our approach makes use of four types of grammatical basis vectors:

1. We use the first principal component of the embeddings of the 30,000 most frequent words. Previous work on word embedding has referred to this as the *common discourse vector*, or c_0, and has shown that this vector encodes words that appear commonly in all contexts, such as 'the'.

2. We take the mean of all vectors of capitalized words and use this as a grammatical basis vector to represent capitalization.

3. For a variety of parts-of-speech, we use the mean vectors for words with that part-of-speech (POS). Specifically, we encode a vector for each of the following: nouns, verbs, adjectives, adverbs, and numbers.

4. We create mean vector differences for the following grammatical concepts: the relationship between singular and plural nouns, the relationship between present-tense verbs and their present participle form, and the relationship between present-tense verbs and their past-tense forms. For each of these relationships, we manually collect approximately 50 example word pairs that fit that relationship. We manually filter for word pairs where either the grammatical relationship does not change the form of the word (i.e., 'deer') or for word pairs where the grammatical change is likely to produce a more complicated change in meaning (i.e., 'math' and 'maths'). We average the differences between pairs of each relationship type and use it as the vector for that relationship.

The choice and construction of these grammatical basis vectors is highly arbitrary, and different grammatical basis vectors could easily be used in different applications or in follow up work.

B Implementation

B.1 Comparison to Faruqui et al.

To compare to the sparse coding approach of Faruqui et al., we use their publicly available implementation with the following settings: We use the same input vectors without preprocessing, a dimensionality of 3000, L_2 regularization penalty $\tau = 10^{-5}$, as suggested in their paper, and various L_1 regularization penalties (λ).

B.2 Word Intrusion Task Implementation

C Randomly Selected Word Representations

We randomly select 25 words and display their complete sparse vector representations here:

$$
\begin{aligned}
\text{carbon} = {} & 0.79 * \text{nitrogen} \\
& - 0.38 * \text{CAPITALIZATION} + 0.3 * \text{fossil} \\
& - 0.21 * \text{POS-NOUN} + 0.16 * \text{POS-ADJ} \\
& + 0.14 * \text{C0} - 0.14 * \text{PAST-TENSE} \\
& + 0.13 * \text{wood} + 0.11 * \text{global} \\
& + 0.1 * \text{atoms} - 0.095 * \text{POS-ADV} \\
& + 0.092 * \text{aluminum} \\
& - 0.078 * \text{PLURAL-NOUN} \\
& + 0.073 * \text{greenhouse} \\
& - 0.072 * \text{POS-PROPN} \\
& - 0.048 * \text{POS-VERB} + 0.046 * \text{forestry} \\
& + 0.03 * \text{PARTICIPLE} + 0.017 * \text{sink} \\
& + 0.012 * \text{POS-NUM}
\end{aligned}
$$

$$
\begin{aligned}
\text{reefs} = {} & 0.68 * \text{islands} \\
& - 0.66 * \text{CAPITALIZATION} + 0.4 * \text{C0} \\
& + 0.35 * \text{PLURAL-NOUN} \\
& + 0.28 * \text{POS-VERB} \\
& + 0.25 * \text{rocks} \\
& + 0.19 * \text{dredging} \\
& + 0.18 * \text{oysters} + 0.12 * \text{POS-ADJ} \\
& + 0.096 * \text{POS-NUM} + 0.096 * \text{POS-ADV} \\
& + 0.089 * \text{POS-PROPN} + 0.086 * \text{tropical} \\
& + 0.075 * \text{underwater} + 0.068 * \text{dunes} \\
& + 0.063 * \text{seas} + 0.06 * \text{diver} \\
& - 0.058 * \text{PAST-TENSE} \\
& + 0.042 * \text{sandstone} \\
& - 0.025 * \text{demon} + 0.02 * \text{marine} \\
& - 0.019 * \text{PARTICIPLE} \\
& - 0.014 * \text{POS-NOUN} \\
& - 0.012 * \text{french} - 0.0041 * \text{witches}
\end{aligned}
$$

$$\begin{aligned}\text{Coulson} = {} & 0.85 * \text{hacking} \\ & + 0.72 * \text{C0} + 0.59 * \text{POS-PROPN} \\ & + 0.25 * \text{CAPITALIZATION} \\ & - 0.23 * \text{POS-ADJ} \\ & - 0.19 * \text{POS-NOUN} + 0.17 * \text{butler} \\ & - 0.17 * \text{southern} + 0.15 * \text{POS-VERB} \\ & - 0.14 * \text{website} - 0.12 * \text{com} \\ & - 0.12 * \text{roaring} + 0.1 * \text{solicitors} \\ & - 0.094 * \text{POS-ADV} + 0.074 * \text{oats} \\ & - 0.068 * \text{cathedral} - 0.064 * \text{PARTICIPLE} \\ & + 0.061 * \text{inquiry} - 0.06 * \text{dances} \\ & - 0.056 * \text{fan} + 0.042 * \text{POS-NUM} \\ & - 0.029 * \text{provinces} - 0.029 * \text{finals} \\ & - 0.02 * \text{dance} - 0.017 * \text{waters} \\ & - 0.013 * \text{tango} - 0.013 * \text{shame} \\ & - 0.012 * \text{PAST-TENSE} \\ & - 0.005 * \text{PLURAL-NOUN}\end{aligned}$$

$$\begin{aligned}\text{environmental} = {} & 0.43 * \text{sustainability} \\ & + 0.43 * \text{economic} + 0.38 * \text{POS-ADJ} \\ & - 0.3 * \text{CAPITALIZATION} - 0.27 * \text{POS-VERB} \\ & + 0.27 * \text{regulatory} - 0.2 * \text{PAST-TENSE} \\ & + 0.18 * \text{biological} + 0.17 * \text{campaigner} \\ & + 0.17 * \text{POS-NUM} + 0.14 * \text{thermal} \\ & - 0.14 * \text{POS-ADV} - 0.12 * \text{POS-NOUN} \\ & + 0.1 * \text{health} - 0.1 * \text{C0} \\ & + 0.087 * \text{PARTICIPLE} + 0.087 * \text{POS-PROPN} \\ & - 0.084 * \text{PLURAL-NOUN} + 0.073 * \text{outdoor} \\ & + 0.055 * \text{chemical} + 0.0055 * \text{cultural}\end{aligned}$$

$$\begin{aligned}\text{roundabout} = {} & 0.72 * \text{bypass} \\ & + 0.4 * \text{roadway} - 0.28 * \text{CAPITALIZATION} \\ & + 0.22 * \text{plaza} - 0.16 * \text{PLURAL-NOUN} \\ & + 0.11 * \text{POS-ADJ} + 0.11 * \text{clumsy} \\ & + 0.1 * \text{airfield} + 0.088 * \text{POS-ADV} \\ & - 0.08 * \text{biological} + 0.079 * \text{C0} \\ & + 0.051 * \text{POS-NOUN} + 0.043 * \text{PAST-TENSE} \\ & + 0.039 * \text{PARTICIPLE} + 0.028 * \text{caravan} \\ & + 0.025 * \text{ironic} + 0.021 * \text{POS-VERB} \\ & - 0.021 * \text{POS-NUM} - 0.0038 * \text{POS-PROPN} \\ & + 0.003 * \text{nonsensical}\end{aligned}$$

$$\begin{aligned}\text{Churchill} = {} & 0.84 * \text{wartime} \\ & + 0.6 * \text{C0} + 0.41 * \text{CAPITALIZATION} \\ & + 0.4 * \text{quotation} + 0.38 * \text{POS-PROPN} \\ & + 0.36 * \text{statesman} - 0.21 * \text{PARTICIPLE} \\ & - 0.14 * \text{astronomer} - 0.14 * \text{POS-NOUN} \\ & - 0.11 * \text{POS-ADJ} - 0.1 * \text{PAST-TENSE} \\ & + 0.082 * \text{POS-VERB} + 0.078 * \text{POS-NUM} \\ & + 0.064 * \text{POS-ADV} + 0.045 * \text{advising} \\ & - 0.025 * \text{architectures} + 0.022 * \text{PLURAL-NOUN} \\ & + 0.017 * \text{pint} + 0.013 * \text{fascism}\end{aligned}$$

$$\begin{aligned}\text{Hub} = {} & 0.49 * \text{bustling} \\ & + 0.47 * \text{C0} + 0.4 * \text{portal} \\ & + 0.39 * \text{infrastructure} + 0.32 * \text{POS-NOUN} \\ & + 0.31 * \text{CAPITALIZATION} + 0.31 * \text{central} \\ & - 0.13 * \text{POS-PROPN} - 0.1 * \text{PLURAL-NOUN} \\ & + 0.069 * \text{outage} + 0.068 * \text{centre} \\ & + 0.061 * \text{POS-NUM} + 0.058 * \text{connectivity} \\ & + 0.058 * \text{PARTICIPLE} - 0.057 * \text{POS-ADJ} \\ & + 0.043 * \text{PAST-TENSE} - 0.043 * \text{POS-VERB} \\ & + 0.027 * \text{POS-ADV}\end{aligned}$$

$$\begin{aligned}\text{resident} = {} & 0.54 * \text{citizens} \\ & + 0.49 * \text{native} + 0.37 * \text{visiting} \\ & - 0.19 * \text{PLURAL-NOUN} + 0.12 * \text{PAST-TENSE} \\ & + 0.11 * \text{caretaker} - 0.099 * \text{CAPITALIZATION} \\ & - 0.094 * \text{C0} + 0.082 * \text{PARTICIPLE} \\ & + 0.082 * \text{ward} + 0.077 * \text{POS-NOUN} \\ & - 0.039 * \text{POS-ADV} + 0.022 * \text{proprietor} \\ & + 0.022 * \text{POS-VERB} - 0.0065 * \text{POS-NUM} \\ & + 0.0045 * \text{POS-PROPN} + 0.0036 * \text{POS-ADJ}\end{aligned}$$

backers $= 0.64 *$ sponsors
$+ 0.4 *$ POS-NOUN $- 0.4 *$ CAPITALIZATION
$+ 0.33 *$ advocates $+ 0.28 *$ PLURAL-NOUN
$+ 0.19 *$ POS-PROPN $+ 0.18 *$ businessman
$+ 0.18 *$ businessmen $+ 0.16 *$ fans
$- 0.15 *$ POS-ADJ $+ 0.12 *$ PARTICIPLE
$+ 0.12 *$ PAST-TENSE $- 0.12 *$ POS-ADV
$+ 0.092 *$ whose $+ 0.082 *$ opposition
$+ 0.065 *$ POS-VERB $+ 0.056 *$ candidacy
$+ 0.055 *$ touted $+ 0.047 *$ startups
$- 0.024 *$ POS-NUM $+ 0.024 *$ rebels
$+ 0.014 *$ reformist $+ 0.013 *$ investment
$- 0.002 *$ C0

re-add $= -0.65 *$ PARTICIPLE
$- 0.47 *$ POS-NUM $+ 0.44 *$ deleted
$+ 0.43 *$ POS-VERB $+ 0.41 *$ cruft
$- 0.41 *$ C0 $- 0.3 *$ PAST-TENSE
$+ 0.28 *$ section $+ 0.19 *$ CAPITALIZATION
$+ 0.17 *$ categorization $+ 0.16 *$ unblock
$+ 0.15 *$ POS-ADV $+ 0.11 *$ POS-PROPN
$+ 0.098 *$ reversion $- 0.09 *$ POS-ADJ
$+ 0.09 *$ POS-NOUN $+ 0.061 *$ inserting
$+ 0.046 *$ reference $+ 0.043 *$ sourcing
$+ 0.034 *$ template $+ 0.027 *$ encyclopedic
$+ 0.013 *$ modify $- 0.0088 *$ battleship
$- 0.0071 *$ cow $+ 0.006 *$ PLURAL-NOUN

rudimentary $= 0.84 *$ basics
$- 0.65 *$ C0 $+ 0.49 *$ POS-ADJ
$- 0.41 *$ POS-VERB $+ 0.41 *$ apparatus
$- 0.36 *$ POS-NOUN $+ 0.35 *$ improvised
$+ 0.15 *$ POS-ADV $+ 0.099 *$ CAPITALIZATION
$+ 0.072 *$ PARTICIPLE $+ 0.069 *$ PLURAL-NOUN
$+ 0.062 *$ POS-NUM $+ 0.059 *$ develop
$+ 0.05 *$ PAST-TENSE $+ 0.043 *$ POS-PROPN

visuals $= 0.47 *$ cinematography
$- 0.47 *$ CAPITALIZATION $+ 0.29 *$ evocative
$+ 0.25 *$ multimedia $+ 0.21 *$ videos
$+ 0.19 *$ POS-NOUN $+ 0.15 *$ PLURAL-NOUN
$+ 0.14 *$ POS-PROPN $+ 0.12 *$ hallucinations
$+ 0.11 *$ awesome $+ 0.1 *$ PARTICIPLE
$+ 0.08 *$ video $- 0.079 *$ POS-VERB
$+ 0.076 *$ sounds $+ 0.076 *$ POS-ADJ
$+ 0.076 *$ slick $+ 0.075 *$ POS-ADV
$+ 0.066 *$ C0 $+ 0.062 *$ dazzling
$+ 0.052 *$ colorful $+ 0.044 *$ interactive
$+ 0.027 *$ jarring $+ 0.019 *$ visualization
$+ 0.0047 *$ PAST-TENSE $+ 0.00025 *$ POS-NUM

admire $= 0.73 *$ admirable
$- 0.66 *$ PARTICIPLE $- 0.65 *$ C0
$+ 0.31 *$ magnificent $+ 0.23 *$ CAPITALIZATION
$+ 0.16 *$ criticize $+ 0.16 *$ POS-NOUN
$- 0.16 *$ PAST-TENSE $+ 0.14 *$ loves
$+ 0.1 *$ POS-PROPN $+ 0.1 *$ beauty
$- 0.098 *$ POS-NUM $- 0.068 *$ PLURAL-NOUN
$+ 0.066 *$ devotion $- 0.061 *$ POS-ADV
$- 0.058 *$ POS-ADJ $+ 0.039 *$ openness
$+ 0.02 *$ charming $- 0.00015 *$ POS-VERB

Conflict $= 0.61 *$ POS-NOUN
$- 0.49 *$ POS-PROPN $+ 0.44 *$ warfare
$+ 0.4 *$ escalation $+ 0.4 *$ peace
$+ 0.36 *$ C0 $+ 0.24 *$ guideline
$+ 0.23 *$ CAPITALIZATION $+ 0.21 *$ PARTICIPLE
$+ 0.19 *$ ethnic $- 0.19 *$ POS-VERB
$+ 0.16 *$ resolved $+ 0.12 *$ POS-NUM
$- 0.099 *$ PLURAL-NOUN $+ 0.078 *$ PAST-TENSE
$+ 0.07 *$ divergence $+ 0.065 *$ geopolitical
$- 0.05 *$ stationary $+ 0.038 *$ POS-ADJ
$- 0.032 *$ shops $+ 0.03 *$ polarized
$- 0.012 *$ POS-ADV

hitter $= -0.54 *$ CAPITALIZATION
$+ 0.45 *$ C0 $- 0.42 *$ PLURAL-NOUN
$+ 0.42 *$ shortstop $+ 0.36 *$ designated
$+ 0.32 *$ batting $+ 0.3 *$ POS-VERB
$+ 0.21 *$ POS-NOUN $+ 0.18 *$ POS-ADV
$+ 0.17 *$ pitchers $+ 0.17 *$ pitcher
$+ 0.14 *$ catcher $- 0.12 *$ PARTICIPLE
$- 0.1 *$ POS-NUM $- 0.096 *$ inane
$+ 0.087 *$ guy $+ 0.073 *$ POS-PROPN
$+ 0.048 *$ exert $- 0.014 *$ PAST-TENSE
$+ 0.0071 *$ outs $- 0.0064 *$ POS-ADJ
$+ 0.0019 *$ swings

fence $= 0.52 *$ wire
$+ 0.43 *$ gates $- 0.41 *$ CAPITALIZATION
$+ 0.35 *$ yard $- 0.32 *$ PLURAL-NOUN
$+ 0.21 *$ shrubs $+ 0.14 *$ barn
$+ 0.14 *$ ditch $+ 0.09 *$ POS-VERB
$+ 0.085 *$ side $- 0.07 *$ PARTICIPLE
$- 0.052 *$ POS-ADJ $+ 0.042 *$ POS-NUM
$- 0.032 *$ POS-PROPN $- 0.02 *$ PAST-TENSE
$- 0.012 *$ C0 $+ 0.0068 *$ nailed
$- 0.0047 *$ POS-ADV $+ 0.00013 *$ POS-NOUN

1978 $= 0.97 *$ 1970s
$- 0.89 *$ POS-ADJ $- 0.6 *$ POS-PROPN
$+ 0.49 *$ POS-NUM $- 0.42 *$ POS-NOUN
$+ 0.21 *$ C0 $- 0.18 *$ PLURAL-NOUN
$- 0.12 *$ POS-ADV $- 0.081 *$ PARTICIPLE
$- 0.073 *$ CAPITALIZATION
$+ 0.067 *$ POS-VERB
$+ 0.041 *$ PAST-TENSE $+ 0.039 *$ seventies
$+ 0.026 *$ contends

heroine $= 0.66 *$ hero
$+ 0.35 *$ protagonist
$- 0.34 *$ CAPITALIZATION
$- 0.25 *$ PLURAL-NOUN $+ 0.14 *$ actress
$+ 0.13 *$ girl $+ 0.1 *$ C0
$+ 0.1 *$ PAST-TENSE $+ 0.071 *$ POS-PROPN
$+ 0.07 *$ POS-NOUN $+ 0.063 *$ POS-ADV
$- 0.058 *$ POS-NUM $+ 0.051 *$ protagonists
$- 0.029 *$ POS-VERB $+ 0.026 *$ PARTICIPLE
$+ 0.015 *$ POS-ADJ $+ 0.014 *$ goddess

structure $= 0.91 *$ structures
$- 0.35 *$ CAPITALIZATION
$- 0.25 *$ PLURAL-NOUN
$+ 0.17 *$ structuring $+ 0.16 *$ POS-NOUN
$- 0.085 *$ POS-VERB $- 0.078 *$ PAST-TENSE
$+ 0.05 *$ POS-ADV $- 0.039 *$ POS-ADJ
$- 0.034 *$ POS-PROPN $+ 0.029 *$ POS-NUM
$+ 0.026 *$ structural $+ 0.022 *$ reorganization
$- 0.022 *$ C0 $+ 0.0079 *$ PARTICIPLE

wizards $= 0.65 *$ magic
$+ 0.41 *$ witches
$- 0.41 *$ CAPITALIZATION
$+ 0.36 *$ PLURAL-NOUN
$+ 0.19 *$ POS-NOUN
$- 0.19 *$ POS-NUM $+ 0.15 *$ POS-ADJ
$+ 0.15 *$ tech $+ 0.13 *$ POS-ADV
$+ 0.098 *$ PAST-TENSE $+ 0.09 *$ wannabe
$+ 0.084 *$ dragons $+ 0.084 *$ knights
$+ 0.052 *$ C0 $- 0.036 *$ POS-PROPN
$+ 0.022 *$ PARTICIPLE $+ 0.015 *$ err
$- 0.013 *$ POS-VERB $+ 0.0041 *$ guru

autistic $= 0.49 *$ preschool
$+ 0.37 *$ epilepsy $- 0.35 *$ POS-NOUN
$+ 0.33 *$ POS-ADJ $- 0.25 *$ papal
$+ 0.22 *$ son $+ 0.21 *$ POS-PROPN
$+ 0.16 *$ twins $+ 0.12 *$ PAST-TENSE
$+ 0.12 *$ therapist $+ 0.12 *$ PARTICIPLE
$- 0.1 *$ CAPITALIZATION $+ 0.084 *$ teenage
$+ 0.072 *$ trait $+ 0.069 *$ psychologist
$+ 0.067 *$ behaviors $+ 0.056 *$ kid
$+ 0.054 *$ hospitalized $- 0.053 *$ C0
$+ 0.052 *$ manipulative $- 0.038 *$ POS-NUM
$+ 0.037 *$ granddaughter $- 0.03 *$ rounds
$+ 0.03 *$ PLURAL-NOUN $+ 0.027 *$ campers
$+ 0.026 *$ POS-VERB $+ 0.0092 *$ dementia
$- 0.0054 *$ regional $- 0.0052 *$ POS-ADV
$- 0.0023 *$ sedan

$$
\begin{aligned}
\text{tornado} &= 0.72 * \text{hurricane} \\
&+ 0.49 * \text{C0} - 0.47 * \text{CAPITALIZATION} \\
&+ 0.28 * \text{typhoon} - 0.26 * \text{PLURAL-NOUN} \\
&+ 0.23 * \text{tractor} + 0.21 * \text{POS-VERB} \\
&+ 0.16 * \text{POS-ADJ} + 0.12 * \text{POS-NUM} \\
&+ 0.1 * \text{flattened} - 0.1 * \text{ports} \\
&- 0.097 * \text{opium} + 0.072 * \text{POS-ADV} \\
&+ 0.053 * \text{POS-PROPN} + 0.053 * \text{avalanche} \\
&+ 0.052 * \text{tape} + 0.05 * \text{earthquake} \\
&- 0.045 * \text{colonial} - 0.043 * \text{POS-NOUN} \\
&+ 0.028 * \text{musical} - 0.025 * \text{handsets} \\
&+ 0.014 * \text{terrifying} + 0.013 * \text{PAST-TENSE} \\
&+ 0.013 * \text{occurrences} - 0.0067 * \text{labour} \\
&+ 0.0025 * \text{PARTICIPLE}
\end{aligned}
$$

$$
\begin{aligned}
\text{recycle} &= -0.65 * \text{PARTICIPLE} \\
&+ 0.61 * \text{bin} + 0.49 * \text{rubbish} \\
&- 0.48 * \text{C0} - 0.47 * \text{PAST-TENSE} \\
&+ 0.27 * \text{POS-VERB} + 0.18 * \text{POS-NOUN} \\
&+ 0.15 * \text{utilize} + 0.14 * \text{plastic} \\
&- 0.12 * \text{POS-NUM} + 0.1 * \text{excess} \\
&+ 0.088 * \text{sustainability} + 0.083 * \text{POS-PROPN} \\
&+ 0.066 * \text{aluminum} + 0.042 * \text{gramthesize} \\
&- 0.037 * \text{PLURAL-NOUN} + 0.036 * \text{POS-ADJ} \\
&+ 0.035 * \text{refurbished} + 0.034 * \text{POS-ADV} \\
&+ 0.03 * \text{converter} + 0.026 * \text{nitrogen} \\
&- 0.004 * \text{CAPITALIZATION} + 0.0024 * \text{saving}
\end{aligned}
$$

$$
\begin{aligned}
1852 &= 0.85 * 1800\text{s} \\
&- 0.81 * \text{POS-PROPN} - 0.78 * \text{POS-ADJ} \\
&- 0.61 * \text{POS-NOUN} + 0.45 * \text{POS-NUM} \\
&+ 0.43 * \text{C0} - 0.31 * \text{CAPITALIZATION} \\
&+ 0.29 * \text{renders} + 0.25 * \text{POS-VERB} \\
&- 0.19 * \text{PLURAL-NOUN} + 0.16 * \text{noted} \\
&- 0.15 * \text{PARTICIPLE} + 0.13 * \text{underscored} \\
&- 0.082 * \text{POS-ADV} + 0.063 * \text{insisting} \\
&+ 0.029 * \text{PAST-TENSE}
\end{aligned}
$$

$$
\begin{aligned}
\text{gloom} &= 0.66 * \text{gloomy} \\
&- 0.46 * \text{CAPITALIZATION} - 0.32 * \text{POS-VERB} \\
&+ 0.28 * \text{pessimism} + 0.28 * \text{darkness} \\
&+ 0.15 * \text{PAST-TENSE} - 0.14 * \text{PLURAL-NOUN} \\
&+ 0.14 * \text{POS-NOUN} + 0.12 * \text{PARTICIPLE} \\
&+ 0.11 * \text{POS-NUM} - 0.058 * \text{C0} \\
&- 0.058 * \text{POS-ADV} + 0.052 * \text{misery} \\
&+ 0.051 * \text{POS-PROPN} + 0.037 * \text{slump} \\
&- 0.025 * \text{POS-ADJ}
\end{aligned}
$$