To be Fair: a Case for Cognitively-Inspired Models of Meaning

Simon Preissner
Center for Mind/Brain Sciences
University of Trento
simon.preissner@gmx.de

Aurélie Herbelot
Center for Mind/Brain Sciences &
Dept. of Information Engineering
and Computer Science
University of Trento
aurelie.herbelot@unitn.it

Abstract

In the last years, the cost of Natural Language Processing algorithms has become more and more evident. That cost has many facets, including training times, storage, replicability, interpretability, equality of access to experimental paradigms, and even environmental impact. In this paper, we review the requirements of a 'good' model and argue that a move is needed towards lightweight and interpretable implementations, which promote scientific fairness, paradigmatic diversity, and ultimately foster applications available to all, regardless of financial prosperity. We propose that the community still has much to learn from cognitively-inspired algorithms, which often show extreme efficiency and can 'run' on very simple organisms. As a case study, we investigate the fruit fly's olfactory system as a distributional semantics model. We show that, even in its rawest form, it provides many of the features that we might require from an ideal model of meaning acquisition. [1]

1 Introduction

In recent years, the Natural Language Processing (NLP) community has seen an increase in the popularity of expensive models requiring enormous computational resources to train and run. The cost of such models is multi-faceted. From the point of view of shaping the scientific community, they create a huge gap between researchers in wealthy institutions and those with less resources and they often make replication prohibitive. From the point of view of applicability, they make the end-user dependent on high-tech hardware which they may not afford, or on cloud services which may have problematic privacy side-effects (and are not available to those with poor Internet access). Training such models can often take a long time and extraordinary amounts of energy, generating CO_2 emissions disproportionate to the models' improvements (Strubell et al., 2019). From a pure modelling point of view, finally, complexity often comes with a loss of interpretability, which weakens theoretical insights. Whilst we appreciate that a part of NLP is focused on engineering applications rather than modelling natural language proper, the linguists and cognitive scientists in the community have a duty to provide transparent, explanatory simulations of particular phenomena.

Such considerations call for smaller and more interpretable systems. In this paper, we offer an example investigation into one of the most widely used techniques in NLP: the vectorial representation of word meanings. Our starting point is the set of requirements that should be fulfilled by an ideal model of lexical acquisition, which is expressed in QasemiZadeh et al. (2017): (A) high performance on fundamental lexical tasks, (B) efficiency, (C) low dimensionality for compact storage, (D) amenability to incremental learning, (E) interpretability. As we will show in §2, state-of-the-art systems still fail to integrate all those points. (A-D) are however basic features of humans and animal cognition. It seems, therefore, that we should find inspiration in algorithms from cognitive science, which in turn would allow us to derive interpretability (E) from the clear underpinnings of biological or psychological theories.

We propose that a good place to find appropriate algorithms is the natural world, as many organisms display core cognitive abilities such as incremental learning, generalization or classification, which many NLP systems need to emulate. Such faculties develop in extremely simple systems, which are good contenders for the type of models we advocate here. One success

story from 'algorithmic' cognitive science is based on the neural architecture of the fruit fly's olfactory system, which clusters patterns of chemicals into categories of smells (Stevens, 2015), and has inspired the so-called *Fruit Fly Optimization Algorithm* (Pan, 2011; here: Fruit Fly Algorithm or 'FFA'). The FFA has been implemented as a lightweight neural algorithm that performs random indexing for locality-sensitive hashing (LSH) (Dasgupta et al., 2017). This LSH algorithm has successfully been applied to various tasks, particularly in information retrieval and for data compression (Andoni and Indyk, 2008). As a simple LSH algorithm, the FFA compresses data while preserving the notion of similarity of the original data, which is one of the core mechanisms involved in constructing vector representations of word meaning. To our knowledge, it has however never been taken as the basis for building distributional semantic models from scratch, even though it seems to naturally fulfill a number of requirements of those models.

In the following, we present the FFA and show how it can be adapted to create vector spaces of word meaning (§4). We then apply the FFA in an incremental setup (§5) and assess its worth as a *model*, according to the various criteria highlighted above (§6), including a possible interpretation of the FFA's output.

2 Related work

In Distributional Semantics (DS: Turney and Pantel, 2010; Erk, 2012), the meaning of words is represented by points in a multidimensional space, derived from word co-occurrence statistics. The quality of models usually correlates with the amount of data that is used. With increasing processing resources and larger corpora available, a variety of approaches have been developed in that area (e.g., Bengio et al., 2003; Pennington et al., 2014; Mikolov et al., 2013). State-of-the-art models perform remarkably well and are often a core component of NLP applications. Recent work on DS (e.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) shifts the scope of representations from word meaning to sentence meaning, pushing performance, but also model complexity, even further.

The latest DS techniques yield high performance, but they have multiple shortcomings. First, they require massive amounts of text, followed by computationally intensive procedures involving weighting, dimensionality reduction, complex attention mechanisms etc. The high complexity of most current architectures often comes at the cost of flexibility: once a language model is constructed, any new data requires a re-run of the complete system in order to be incorporated. This makes incrementality unsatisfiable in those frameworks (Sahlgren, 2005; Baroni et al., 2007). Further, architectures themselves have become increasingly complex, at the expense of transparency. We recall that even Word2Vec (W2V: Mikolov et al., 2013), which is a comparatively simple system by today's standards, has attracted a large amount of literature which attempts to explain the effects of various hyperparameters in the model (Levy and Goldberg, 2014; Levy et al., 2015; Gittens et al., 2017). Finally, high-performance DS representations are hardly or not at all interpretable. As a result, much research has been dedicated to producing representations that are intuitively interpretable by humans (Murphy et al., 2012; Luo et al., 2015; Fyshe et al., 2015; Shin et al., 2018). These approaches typically attempt to preserve or reconstruct word labels for the basis of the dimensionality-reduced representations, but they can themselves require intensive procedures. In summary, it becomes apparent that the ideal vector-based semantics model that fulfills all requirements highlighted in our introduction has not yet been found.

The Fruit Fly Algorithm we present here can be related to two existing techniques in computer science: Random Indexing and Locality-Sensitive Hashing. Random Indexing (RI) is a simple and efficient method for dimensionality reduction (cf. Sahlgren, 2005), originally used to solve clustering problems (Kaski, 1998). It is also a less-travelled technique in distributional semantics (Kanerva et al., 2000; QasemiZadeh et al., 2017; QasemiZadeh and Kallmeyer, 2016). Its advocates argue that it fulfills a number of requirements of an ideal vector space construction method, in particular incrementality. As for Locality-Sensitive Hashing (LSH: Slaney and Casey, 2008), it is a way to produce hashes that preserve a notion of distance between points in a space, thus satisfying storage efficiency whilst maintaining the spatial configuration of a representation. A comparison of various hash functions for LSH, including RI, is provided by Paulevé

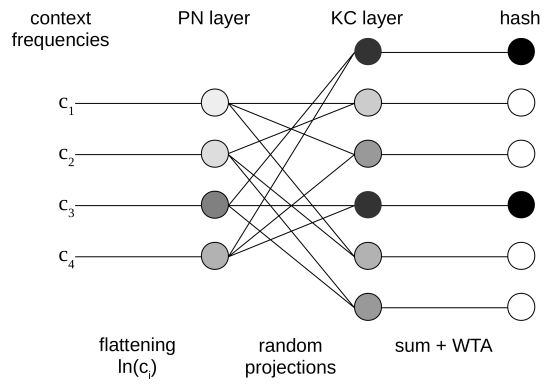

Figure 1: Schematic of the adapted FFA, with input size $m = 4$ and output size $n = 6$ (dense representation: 2). Darker cells correspond to higher activation.

et al. (2010).

3 Data

In the spirit of 'training small', the corpus used for our experiments is a subset of 100M words from the ukWaC corpus (Ferraresi et al., 2008), minimally pre-processed (tokenized and stripped of punctuation signs); this results in a corpus of 87.8M words. Following common practice, we quantitatively evaluate the FFA as a lexical acquisition algorithm by testing it over the MEN similarity dataset (Bruni et al., 2014), which consists of 3000 word pairs (751 unique English words), human-annotated for semantic relatedness.

For our experiments, we compute two co-occurrence count spaces over our corpus, with different context sizes (± 2 and ± 5 around the target). We only consider the 10k most frequent words in the data, ensuring we cover all 751 words in MEN.

4 Model

The Fruitfly Algorithm mimics the olfactory system of the fruit fly, which assigns a pattern of binary activations to a particular smell (i.e., a combination of multiple chemicals), using sparse connections between just two neuronal layers. This mechanism allows the fly to 'conceptualize' its environment and to appropriately react to new smells by relating them to previous experiences. Our implementation of the FFA is an extension of the work of Dasgupta et al. (2017) which allows us to generate a semantic space by hashing each word – as represented by its co-occurrences in a corpus – to a pattern of binary activations.

As in the original implementation, our FFA is a simple feedforward architecture consisting of two layers connected by random projections (Fig. 1). The input layer, the *projection neuron layer* or *PN layer*, consists of m nodes $\{x_1 ... x_m\}$ which encode the raw co-occurrence counts of a target word with a particular context. To satisfy incrementality, m is variable and can grow as the algorithm encounters new data. If a new context is observed, then a node x_{m+1} is recruited to encode that context. A logarithmic function is applied to the input in order to diminish frequency effects of natural languages (Zipf, 1932). This 'flattens' activation across the PN layer, reducing the impact of very frequent words (e.g., stopwords). The second layer (*Kenyon Cell layer* or *KC layer*) consists of n nodes $\{y_1 ... y_n\}$. It is larger than the PN layer and fixed at a constant size (n does not grow). PN and KC are *not* fully connected. Instead, each KC cell receives a constant number of connections from the PN layer, randomly and uniformly allocated. In other words, the mapping from PN to KC is a bipartite connection matrix M so that $M_{ji} = 1$ if x_i is connected to y_j and 0 otherwise. The connectivity of each PN is thus variable, albeit uniformly distributed. The activation function on each KC is simply the sum of the activations of its connected PNs. In the end, hashing is carried out via a winner-takes-all (WTA) procedure that 'remembers' the IDs of a small percentage of the most activated KCs as a compact representation of the word's meaning. So $WTA(y_i) = 1$ if y_i is one of the k top values in y and 0 otherwise.

The FFA's hyperparameters are expressed as a 5-tuple (f, m, n, c, h), where f is the flattening function, m is the size of the PN layer (initially 0), n is the size of the KC layer, c is the number of connections leading to any one KC, and h is the percentage of activated KCs to be hashed.

Note that, since both the connectivity per KC and the size of the KC layer are constant, the overall number of connections is constant. Thus, the expansion mechanism (which increments m) does not create new connections: it randomly selects existing PNs and reallocates connections from those PNs to the new PN. In the reallocation process, we encode a bias towards taking connections from those PNs with the most outgoing connections in order to ensure even connectivity of the PN layer. For example, in a setup with parameters $(f = ln, m = 300, n = 10000, c = 14, h = 8)$,

the average number of connections going out from each PN is $(n \times c)/m = 466.67$: some PNs have 466 connections, some have 467 or more. The next newly encountered word will lead to the creation of x_{301} and the expansion process will reallocate $\lfloor (n \times c)/301 \rfloor = 465$ already existing connections to x_{301}. For this, it will choose PNs with 467 or more connections with a higher probability than those with 466 connections. The parameters after the expansion process are $(f = ln, m = 301, n = 10000, c = 14, h = 8)$.

The expansion of dimensions from the PN layer to the KC layer in combination with random projections can be interpreted as a form of 'zooming' into a concept for a particular target word: multiple context words are randomly projected onto a single KC. If several of these context words are important for the target (i.e., their PNs have high activation), the corresponding KC will be activated in the final hash. We can imagine this process as aggregating dimensions of the original co-occurrence space, thus generating latent features which give different 'views' into the raw data. For example, one might imagine that a random projection from the PNs *beak, bill, bank, wing*, and *feather*, have one KC in common. This KC might be somewhat activated by the PNs *bank* and *bill* in finance contexts, but more crucially, it will consistently be strongly activated for target words related to birds and thus selected for the final hashes of those words. Note that this behaviour lets us backtrack from a dimensionality-reduced representation to the most characteristic contexts for a particular target word, and gives interpretability to the KCs. We will come back to that feature in §6.

5 Experiments and results

In order to characterize the behavior and performance of our incremental FFA, we evaluate the quality of its output vectors against the MEN test set by means of the non-parametric Spearman rank correlation ρ. In order to run the experiments with a sound configuration of the hyperparameters f, n, c, and h, we first perform a grid search, applying various configurations of the FFA to the counts (window size: ± 5) of the 10k most frequent words of a held-out corpus.[2] For this setting, the grid search yields the following optimal configuration:

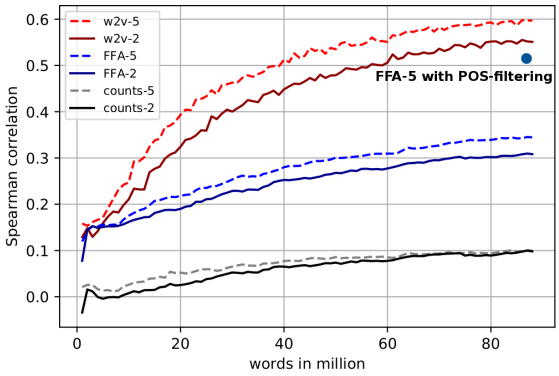

Figure 2: ρ-values of co-occurrence counts, hashed spaces, and Word2Vec models (window sizes $\pm$ 2 (lines) and $\pm$ 5 (dotted)). The blue dot shows the performance on POS-tagged data with FFA-5.

$(f = ln, n = 40000, c = 20, h = 0.08)$; we use this for all further experiments.[3] (The grid search revealed in fact that the factor of expansion $\frac{n}{m}$ is minimally important.)

Next, we *incrementally* generate a raw frequency-count model of the 10k most frequent words of our corpus, parallelly expanding the FFA with every newly encountered word. Every 1M processed words, the aggregated co-occurrences are hashed by the FFA and the corresponding word vectors (i.e., binary hashes) are stored for evaluation. We compare a) the raw frequency space (input to the FFA); b) the final hashes (output of the FFA); c) a separate Word2Vec (W2V) model trained on exactly the same data, using standard hyperparameters and a minimum count set to match the 10k target words of our co-occurrence space. We repeat this experiment for window sizes ± 2 and ± 5.

Figure 2 shows the results of our incremental simulation. For the window size ± 5, we reach ρ = 0.100 for raw counts, ρ = 0.345 for the FFA output, and ρ = 0.600 for W2V. The 2-word-context setup yields very similar results. The FFA hashing thus has a clear and positive effect (+0.245 from 80M words on for the ± 5 setup). The amount of improvement is already large at the beginning of training (+0.136 at 5M words) and slowly increases with corpus size. Results are comparable to W2V for very small corpus sizes, but start lagging behind after around 10M words.

[2] we restricted the grid search and the subsequent experiment setup to a vocabulary of 10k words for more convenient experimentation. The actual FFA potentially has no such limit

[3] The source code of this implementation of the FFA will be released for public use on `git@github.com: SimonPreissner/semantic-fruitfly.git`

6 Discussion

Investigating cognitive algorithm from scratch requires a clear stance on evaluation: we cannot expect a very simple model to beat the performance of heavily-trained systems, but we can require it to give satisfactory results whilst also being a good *model* in the strong sense of the term, that is, simulating all observable features of a given real-world phenomenon. Our discussion keeps this in mind, as we focus on the 'wish list' highlighted in §1.

Performance: hashing increases performance over the raw co-occurrence space by over 20 points overall. The system is however outperformed by W2V after seeing around 10M words. In the spirit of providing a comprehensive evaluation of the modelling power of the FFA, we attempt to pull apart aspects of the learning process that are captured by its very simple algorithm, and those that are not. In other words, which feature results in the large increase over baseline performance? What does the FFA fail to model with respect to W2V? We know that the algorithm generates latent features out of the original space dimensions, encapsulated in each KC. We have tuned the size of the KC layer, so the number of features captured by the FFA should be optimal for our task. We assume that the performance displayed by the algorithm is due to correctly generalizing over contexts. As for its *lack* of performance, we can make hypotheses based on what we know from other DS models. The FFA does not perform any subsampling or weighting of its input data, and the log function we use to minimize the impact of very frequent items is probably too crude to fulfill that purpose. When we informally inspect the performance of the algorithm on a POS-tagged version of our corpus, keeping only verbs, nouns and adjectives in the input and filtering some highly frequent stopwords (punctuation, auxiliaries), we obtain $\rho \approx 0.51$ over the whole corpus,[4] coming close to W2V's performance and thus indicating that indeed, a higher-level 'attention' mechanism could be added to the input layer. (Note that the olfactory system of actual fruit flies only has $\approx$ 50 odorant receptors, which makes it potentially less crucial to successfully suppress large parts of the input.)

Dimensionality: The size of the hashes produced by the FFA is variable; in the experiments, it was set to 3200,[5] which is much larger than the optimal 300-400 dimensions of W2V. However, the hash corresponds to a sparse vector of *integers* and is thus efficiently stored and manipulated. The hyperparameter grid search revealed that the factor of expansion from PN layer to KC layer is much less important than expected, although the expansion is a core characteristic of the FFA and intuitively, its factor should have an effect on performance. This suggests that the FFA does not require inconveniently high-dimensional hash signatures to reach its performance. However, it will take further experiments, especially with larger vocabularies, to fully characterize this behaviour.

Incrementality: the FFA is fully incremental. Note that in our experiments, the W2V space is retrained from scratch after each addition of 1M words to the corpus while the FFA simply increments counts in its stored co-occurrence space. It is also in stark contrast with weighted count-based distributional models which require some global PMI (re-)computation to outperform the raw co-occurrence count vectors.

Time efficiency: our FFA runs without costly learning mechanisms; its two most costly operations are (1) the expansion of the PN layer along with new vocabulary and (2) the projection from PN layer to KC layer. Following Zipf's Law, most new words are encountered within the first few millions of words. As a consequence, the frequency of expansion operations on the PN layer is high at first, but decreases rapidly, resulting in fast scaling to large amounts of text. Hashing is solely dependent on the number of connections per KC and the size of the KC layer (both constant).

Interpretability: the FFA's two-layer architecture allows for uncomplicated backtracking. Each of the activated nodes in a word's hash represents a single KC. The connections of these 'winner' KCs with the PN layer let us reconstruct which context words originally contributed to the largest activations in the KC layer. To illustrate this, we use the hashes obtained at the last iteration of our incremental experiment (based on window ± 5) and identify the $k = 50$ most characteristic PNs for each hash, ignoring stopwords. Table 1 reports the characteristic PNs shared by various sets of input words. For example, for the words *hawk, pi-*

[4] We use the top 4000 dimensions of the co-occurrence matrix, with $n = 16000$, $c = 20$ and $h = 0.08$.

[5] This results from expressing the (n=40k-dimensional) binary vector as the positions of its 1s, which make up $h = 8\%$ of the vector. This yields a much smaller representation of length $n \times h = 3200$.

Hashed Words	Mutual Important Words
hawk, pigeon, parrot	tailed, breasted, black, red, dove
library, collection, museum	collection, national, new, art
beard, wig	man, wearing, long, like, hair
cold, dirty	get, said, war, mind

Table 1: Top PNs for selected sets of words. The importance of a PN for a word is estimated by the number of connections to KCs that are activated in the word's hash (window size $\pm$ 5).

geon, and *parrot* the *tailed, black, breasted, red,* and *dove* PNs are among the most influential, contributing to many of the activated KCs. Similarly, we can connect *beard* to *wig* and *cold* to *dirty*; the shared important words of the latter seem to encode shared collocates (*cold/dirty war, cold/dirty mind, get cold/dirty*).

7 Conclusion

We started this paper suggesting that NLP should explore a different class of algorithms for its most fundamental tasks. We argued that it is worth investigating cognitively-inspired architectures, which may not (yet) perform at state-of-the-art level, but give us insights into potentially more plausible ways to model linguistic faculties in the mind. We also made a case for 'small' and 'fair' systems, in reach of all researchers and end-users.

As illustration, we have explored what the olfactory system of a fruit fly can do for the representation of word meanings. The algorithm is certainly 'fair' in terms of complexity and required resources. Being based on an actual cognitive mechanism, it naturally encodes requirements such as (processing and storage) efficiency. Its simplicity lends itself to incremental learning and interpretability. Performance on a relatedness data set highlights that the raw model successfully captures latent concepts in the data but would probably require an extra attention layer, as indicated by the stronger results obtained on additionally preprocessed data.

We hope to have demonstrated that such study is accessible to all, and actually sheds insights into the minimal components of a model in a way that more complex systems do not achieve. We particularly draw attention to the fact that the interesting behaviour of the fruit fly with respect to interpretability and incrementality makes it a worthy competitor for other distributional models – or at the very least, a source of inspiration.

References

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117.

Marco Baroni, Alessandro Lenci, and Luca Onnis. 2007. Isa meets lara: An incremental word space model for cognitively plausible simulations of semantic learning. In *Proceedings of the workshop on cognitive aspects of computational language acquisition*, pages 49–56.

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

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

Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. 2017. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796.

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

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Alona Fyshe, Leila Wehbe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2015. A compositional and interpretable semantic space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 32–41.

Alex Gittens, Dimitris Achlioptas, and Michael W Mahoney. 2017. Skip-gram- zipf+ uniform= vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–76.

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

Samuel Kaski. 1998. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, pages 413–418. IEEE.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

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

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1687–1692.

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

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

Wen-Tsao Pan. 2011. A new evolutionary computation approach: fruit fly optimization algorithm. In *Proceedings of the conference on digital technology and innovation management*.

Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. 2010. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358.

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Behrang QasemiZadeh and Laura Kallmeyer. 2016. Random positive-only projections: Ppmi-enabled incremental semantic space construction. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 189–198.

Behrang QasemiZadeh, Laura Kallmeyer, and Aurelie Herbelot. 2017. Projection aléatoire non-négative pour le calcul de word embedding. In *24e Conférence sur le Traitement Automatique des Langues Naturelles (TALN)*, pages 109–122.

Magnus Sahlgren. 2005. An introduction to random indexing. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*.

Jamin Shin, Andrea Madotto, and Pascale Fung. 2018. Interpreting word embeddings with eigenvector analysis. *32nd Conference on Neural Information Processing Systems (NIPS 2018), IRASL workshop*.

Malcolm Slaney and Michael Casey. 2008. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal processing magazine*, 25(2):128–131.

Charles F Stevens. 2015. What the flys nose tells the flys brain. *Proceedings of the National Academy of Sciences*, 112(30):9460–9465.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

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

George Kingsley Zipf. 1932. *Selected studies of the principle of relative frequency in language*. Harvard university press.