300-sparsans at SemEval-2018 Task 9: Hypernymy as interaction of sparse attributes

Gábor Berend
Department of Informatics
University of Szeged
Árpád tér 2, H6720 Szeged, Hungary
berendg@inf.u-szeged.hu

Márton Makrai
Institute for Linguistics
Hungarian Academy of Sciences
Benczúr u. 33, H1068 Budapest, Hungary
makrai.marton@nytud.mta.hu

Péter Földiák
Secret Sauce Partners
657 Mission Suite 410,
San Francisco CA 94105
Peter.Foldiak@gmail.com

Abstract

This paper describes 300-sparsans' participation in SemEval-2018 Task 9: *Hypernym Discovery*, with a system based on sparse coding and a formal concept hierarchy obtained from word embeddings. Our system took first place in subtasks *(1B) Italian* (*all* and *entities*), *(1C) Spanish entities*, and *(2B) music entities*.

1 Introduction

Natural language phenomena are extremely sparse by their nature, whereas continuous word embeddings employ dense representations of words. Turning these dense representations into a much sparser form can help in focusing on most salient parts of word representations (Faruqui et al., 2015; Berend, 2017; Subramanian et al., 2018).

Sparsity-based techniques often involve the coding of a large number of signals over the same dictionary (Rubinstein et al., 2008). Sparse, overcomplete representations have been motivated in various domains as a way to increase separability and interpretability (Olshausen and Field, 1997) and stability in the presence of noise.

Non-negativity has also been argued to be advantageous for interpretability (Faruqui et al., 2015; Fyshe et al., 2015; Arora et al., 2016). As Subramanian et al. (2018) illustrates this in the language domain, where sparse features are interpreted as lexical attributes, "to describe the city of Pittsburgh, one might talk about phenomena typical of the city, like erratic weather and large bridges. It is redundant and inefficient to list negative properties, like the absence of the Statue of Liberty". Berend (2018) utilizes non-negative sparse coding for word translation by training sparse word vectors for the two languages such that coding bases correspond to each other.

Here we apply sparse feature pairs to hypernym extraction. The role of an attribute pair $\langle i, j\rangle \in \phi(q) \times \phi(h)$ (where q is the query word, h is the hypernym candidate, and $\phi(w)$ is the index of a non-zero component in the sparse representations of w) is similar to *interaction terms* in regression, see section 2 for details.

Sparse representation is related to hypernymy in various natural ways. One of them is through *Formal concept Analysis (FCA)*. The idea of acquiring concept hierarchies from a text corpus with the tools of Formal concept Analysis (FCA) is relatively new (Cimiano et al., 2005). Our submissions experiment with formal concept analysis tool by Endres et al. (2010). See the next section for a description of formal concept lattices, and how hypernyms can be found in them.

Another natural formulation is related to *hierarchical sparse coding* (Zhao et al., 2009), where trees describe the order in which variables "enter the model" (i.e., take non-zero values). A node may take a non-zero value only if its ancestors also do: the dimensions that correspond to top level nodes should focus on "general" meaning components that are present in most words. Yogatama et al. (2015) offer an implementation that is efficient for gigaword corpora. Exploiting the correspondence between the variable tree and the hypernym hierarchy offers itself as a natural choice.

The task (Camacho-Collados et al., 2018) evaluated systems on their ability to extract hypernyms for query words in five subtasks (three languages, English, Italian, and Spanish, and two domains,

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 928–934
New Orleans, Louisiana, June 5–6, 2018. ©2018 Association for Computational Linguistics

medical and music). Queries have been categorized as concepts or entities. Results were reported for each category separately as well as in combined form, thus resulting in 5×3 combinations. Our system took first place in subtasks *(1B) Italian* (*all* and *entities*), *(1C) Spanish entities*, and *(2B) music entities*. Detailed results for our system appear in section 3. Our source code is available online[1].

1.1 Formal concept analysis

Formal concept Analysis (FCA) is the mathematization of *concept* and conceptual hierarchy (Ganter and Wille, 2012; Endres et al., 2010). In FCA terminology, a *context* is a set of *objects* $\mathcal{O}$, a set of *attributes* $\mathcal{A}$, and a binary incidence relation $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{A}$ between members of $\mathcal{O}$ and $\mathcal{A}$. In our application, $\mathcal{I}$ associates a word $w \in \mathcal{O}$ to the indices of its non-zero sparse coding coordinates $i \in \mathcal{A}$. FCA finds formal *concepts*, pairs $\langle O, A\rangle$ of object sets and attribute sets ($O \subseteq \mathcal{O}, A \subseteq \mathcal{A}$) such that A consists of the shared attributes of objects in O (and no more), and O consists of the objects in $\mathcal{O}$ that have all the attributes in A (and no more). (There is a closure-operator related to each FCA context, for which O and A are closed sets iff $\langle O, A\rangle$ is a concept.) O is called the extent and A is the intent of the concept.

There is an order defined in the context: if $\langle A_1, B_1\rangle$ and $\langle A_2, B_2\rangle$ are concepts in C, $\langle A_1, B_1\rangle$ is a *subconcept* of $\langle A_2, B_2\rangle$ if $A_1 \subseteq A_2$ which is equivalent to $B_1 \supseteq B_2$. The concept order forms a lattice. The smallest concept whose extent contains a word is said to *introduce* the object. We expect that h will be a hypernym of q iff $n(q) \leq n(h)$ where $n(w)$ denotes the node in the concept lattice that introduces w.

The closedness of extents and intents has an important structural consequence. Adding attributes to $\mathcal{A}$ (e.g. responses of additional neurons) will very probably grow the model. However, the original concepts will be embedded as a substructure in the larger lattice, with their ordering relationships preserved.

2 Our approach

Now we describe our system that is based on sparse non-negative word representations and FCA besides more traditional features.

We use the popular skip-gram (SG) approach (Mikolov et al., 2013) to train $d = 100$ dimensional dense distributed word representations for each sub-corpus. The word embeddings are trained over the text corpora provided by the shared task organizers with the default training parameters of `word2vec` (w2v), i.e. a window size of 10 and 25 negative samples for each positive context.

We derived *multi-token units* by relying on the `word2phrase` software accompanying the w2v toolkit. An additional source for identifying multi-token units in the training corpora was the list of potential hypernyms released for each subtask by the shared task organizers.

Given the dense embedding matrix $W_x \in \mathbb{R}^{d \times |V_x|}$, for some subcorpus of the shared task $x \in \{1A, 1B, 1C, 2A, 2B\}$, where $|V_x|$ is the size of the vocabulary and d is set to 100. As a subsequent step, we turn W_x into *sparse word vectors* akin to Berend (2017) by solving for

$$\min_{D \in \mathcal{C}, \alpha \in \mathbb{R}_{\geq 0}} \|D\alpha - W_x\|_F + \lambda\|\alpha\|_1, \qquad (1)$$

where $\mathcal{C}$ refers to the convex set of $\mathbb{R}^{d \times k}$ matrices consisting of d-dimensional columns vectors with norm at most 1, and α contains the sparse coefficients for the elements of the vocabulary. The only difference compared to Berend (2017) is that here we ensure a non-negativity constraint over the elements of α.

For the elements of the vocabulary we ran the *formal concept analysis* tool of Endres et al. (2010)[2]. In order to keep the size of the DAG outputted by the FCA algorithm manageable, we only included the query words and those hypernyms in the analysis which occur in the training dataset for the corpora. As we will see in the next section, this restriction turns out to be very useful.

Next, we determine a handful of features for a pair of expressions (q, h) consisting of a query q and its potential hypernym h. Table 1 provides an overview of the features employed for a pair (q, h). We denote with $\mathbf{q}$ and $\mathbf{h}$ the 100-dimensional dense vectorial representations of q and h. Additionally, we denote with Q and H the sequence of tokens constituting the query and hypernym phrases. Finally, we refer to the set of basis vectors (in the FCA terminology, attributes)

[1] `https://github.com/begab/fca_hypernymy`

[2] `www.compsens.uni-tuebingen.de/pub/pages/personals/3/concepts.py`

Core feature name	
cosine	$\frac{\mathbf{q}^\intercal\mathbf{h}}{\|\mathbf{q}\|_2\|\mathbf{h}\|_2}$
difference	$\|\mathbf{q}-\mathbf{h}\|_2$
normRatio	$\frac{\|\mathbf{q}\|_2}{\|\mathbf{h}\|_2}$
qureyBeginsWith	$Q[0] = h$
queryEndsWith	$Q[-1] = h$
hasCommonWord	$Q \cap H \neq \emptyset$
sameFirstWord	$Q[0] = H[0]$
sameLastWord	$Q[-1] = H[-1]$
logFrequencyRatio	$\log_{10}\frac{count(q)}{count(h)}$
isFrequentHypernym[3]	$c \in MF_{50}(q.type)$
sameConcept	$n(h) = n(q)$
parent	$n(q) \prec n(h)$
child	$n(h) \prec n(q)$
overlappingBasis	$\phi(q) \cap \phi(h) \neq \emptyset$
sparseDifference$_{q\setminus h}$	$\lvert\phi(q) - \phi(h)\rvert$
sparseDifference$_{h\setminus q}$	$\lvert\phi(h) - \phi(q)\rvert$
attributePair$_{ij}$	$\langle i, j\rangle \in \phi(q) \times \phi(h)$

Table 1: The features employed in our classifier. $MF_{50}(q.type)$ refers to the set of top-50 most frequent hypernyms for a given query type.

which are assigned non-zero weights in the reconstruction of the vectorial representation of q and h as $\phi(q)$ and $\phi(h)$. It is also considered as a feature (`isFrequentHypernym`) whether a particular candidate hypernym h belongs to the top-50 most frequent hypernyms for the category of q (i.e. concept or entity). Modeling the two categories separately played an important role in the success of our systems.

Three additional features are defined for incorporating the concept lattice output by FCA. With $n(w)$ denoting the concept that introduces w, i.e. the most specific location within the DAG for w, our features indicate whether $n(q)$ (1) coincides with that of h, (2) is the parent (immediate successor) for that of h, or (3) is the child (immediate predictions) for that of h. Parents, and even the inverse relation, proved to be more predictive than the conceptually motivated $q \leq h$. In Table 1, $n_1 \prec n_2$ denotes that n_1 is an immediate predecessor of n_2. We will see in post-evaluation ablation experiments, where we refer to the above three features as the *FCA* features, that they were not useful in our submissions.

The `attributePair`$_{ij}$s above, our most important features, are indicator features for every possible interaction term between the sparse coefficients in α. That means that for a pair of words (q, h) we defined $\phi(q) \times \phi(h)$, i.e. candidates get assigned with the Cartesian product derived from the indices of the non-zero coefficients in α. Note that this feature template induces k^2 features, with k being the number of basis vectors introduced in the dictionary matrix D according to Eq. 1.

In order to rank potential hypernym candidates over the test set we trained a *logistic regression* classifier for concepts and entities utilizing the `sklearn` package (Pedregosa et al., 2011)[4] with the regularization parameter defaulting to 1.0.

For each appropriate (q, h) pair of words for which h is a hypernym of q, we generated a number of *negative samples* (q, h'), such that the training data does not include h' as a valid hypernym for q. For a given query q, belonging to either of the *concept* or *entity* category, we sampled h' from those hypernyms which were included as a valid hypernym in the training data with respect to some $q' \neq q$ query phrase.

When making predictions for the hypernyms of a query, we relied on our query type sensitive logistic regression model to determine the ranking of the hypernym candidates. In our official submission we treated such phrases to rank which were included in the training data for being a proper hypernym at least once.

After the appropriate model ranked the hypernym candidates, we selected the top 15 ranked candidates and applied a *post-ranking* heuristic over them, i.e. reordered them according to their background frequency from the training corpus. Our assumption here is that more frequent words tend to refer to more general concepts and more general hypernymy relations potentially tend to be more easily detectable than more specialized ones.

3 Results

3.1 Our submissions

Our submissions were based on $k = 200$ dimensional sparse vectors computed from unit-normed 100-dimensional dense vectors with $\lambda = .3$. The sum of the two dimensions motivates our group name. For training the regression model with negative samples, 50 false hypernyms were sampled for each query q in the training dataset. One of our

[3] At submission time, this feature did not work properly.

[4] `scikit-learn.org`

		without attribute pairs						with attribute pairs					
		MAP	MRR	P@1	P@3	P@5	P@15	MAP	MRR	P@1	P@3	P@5	P@15
1A	offic	8.6	18.0	13.0	8.9	8.2	7.9	8.9	19.4	14.9	9.3	8.6	8.1
1A	reprd	9.07	18.7	13.5	9.4	8.8	8.5	9.2	19.9	14.9	9.5	8.7	8.4
1B	offic	9.4	19.9	13.2	9.5	9.3	8.8	12.1	25.1	17.6	12.9	11.7	11.2
1B	reprd	9.2	19.5	12.8	8.9	8.9	8.7	12.8	26.7	18.9	13.6	12.4	11.9
1C	offic	12.5	25.9	16.6	13.6	12.6	11.5	17.9	37.6	27.8	19.7	17.1	16.3
1C	reprd	12.9	26.0	16.2	13.9	13.0	11.9	18.3	38.4	28.5	20.2	17.4	16.6
2A	offic	15.0	32.2	24.8	17.7	15.8	11.6	20.8	40.6	31.6	23.5	21.4	17.1
2A	reprd	15.1	32.4	24.4	18.0	16.2	11.8	21.5	43.7	35.6	25.3	21.8	17.0
2B	offic	19.1	36.7	27.2	23.0	20.1	15.4	29.5	46.4	33.0	31.9	28.9	27.7
2B	reprd	21.5	40.9	29.6	25.6	22.1	18.0	30.4	46.8	33.8	31.8	29.5	28.9

Table 2: Our submissions results: `offic`ial and those that can be `reprod`uced with the code in the project repo (with the *isFrequentHypernym* feature being turned off).

	MAP	MRR	P@1	P@3	P@5	P@10
1A	9.8	22.6	19.8	10.0	9.0	8.6
1A	8.8	21.4	19.8	8.9	7.8	7.5
1B	8.9	21.2	17.1	9.1	8.3	7.9
1B	7.8	19.4	17.1	8.3	6.8	6.5
1C	16.4	33.3	24.6	17.5	16.1	14.9
1C	12.2	29.8	24.6	12.0	11.3	11.0
2A	29.0	35.9	32.6	34.3	34.2	21.7
2A	28.9	35.8	32.6	34.3	34.2	21.4
2B	40.2	58.8	50.6	44.6	40.3	35.5
2B	33.3	51.5	36.2	40.1	35.8	28.4

Table 3: Baseline results, most frequent training hypernyms. We (upper) consider the most frequent hypernym in the given query type (concept or entity). For comparison, we also show the `MFH` baseline provided by the organizers (lower) that is based on the most frequent hypernyms in general.

	Train		Test	
1A	975(4)	0.41%	1055(4)	0.38%
1B	709(1)	0.14%	767(2)	0.26%
1C	776(2)	0.26%	625(2)	0.32%
2A	442(58)	11.60%	433(67)	13.40%
2B	366(21)	5.43%	341(17)	4.75%

(a) concept

	Train		Test	
1A	379(142)	27.26%	344(99)	22.35%
1B	249(41)	14.14%	205(26)	11.26%
1C	184(38)	17.12%	328(45)	12.06%
2A	0(0)	—	0(0)	—
2B	79(34)	30.09%	102(40)	28.17%

(b) entity

Table 4: Number of in-vocabulary (and out-of-vocabulary, OOV) queries per query type. The ratio of the latter is also shown.

submissions involved attribute pairs, the other not. Both submissions used the conceptually motivated but practically harmful FCA-based features.

Table 2 shows submission results. The figures that can be reproduced with the code in the project repo (`reprd`) is slightly different from our official submissions (`offic`) for two reasons: because the implementation of `isFreqHyp` contained a bug, and because of the natural randomness in negative sampling. For reproducibility, we report result without the `isFreqHyp` feature. The randomness introduced by negative sampling is now factored out by random seeding.

3.2 Query type sensitive baselining

Our submission with attribute pairs achieved first place in categories (1B) Italian (all and entities), (1C) Spanish entities, and (2B) music entities. This is in part due to our good choice of a fallback solution in the case of OOV queries: we applied a category-sensitive baseline returning the most frequent train hypernym in the corresponding query type (concept or entity). Table 4 shows how frequently we had to rely on this fallback, and Table 3 shows the corresponding pure baseline results.

		candidate filtering off						candidate filtering on					
k	ns	MAP	MRR	P@1	P@3	P@5	P@15	MAP	MRR	P@1	P@3	P@5	P@15
200	50	6.5	14.9	13.1	7.4	6.1	5.5	12.1	25.4	18.9	12.9	11.6	10.9
200	all	6.9	15.8	14.1	7.6	6.3	5.8	13.0	27.1	19.9	14.2	12.5	11.8
300	50	6.9	15.8	13.9	7.6	6.4	5.9	12.1	25.7	19.5	13.0	11.5	11.0
300	all	8.0	17.8	15.4	8.9	7.4	6.8	13.5	28.0	21.1	14.5	12.9	12.3
1000	50	9.0	20.0	17.2	9.8	8.3	7.7	13.3	**28.1**	**21.3**	13.8	12.6	12.3
1000	all	11.6	26.1	22.5	12.5	10.8	10.0	**13.6**	27.2	19.4	**13.9**	**13.2**	**12.8**

Table 5: Post evaluation results on the 1A dataset investigating the effect of various hyperparameter choices. k and ns denotes the number of basis vectors and negative samples generated during training per each positive (q, h) pair. Best results obtained for each metric are marked as bold.

		MAP	MRR	P@1	P@3	P@5	P@15
off	off	10.3	21.3	15.0	10.6	10.1	9.6
off	on	10.1	21.1	14.9	10.5	9.9	9.5
on	off	12.1	25.4	18.9	12.9	11.6	10.9
on	on	12.1	25.3	18.7	13.0	11.6	11.0

Table 6: Ablation experiments, on the 1A dataset with $k = 200, ns = 50$ (and the implementation of `isFreqHyp` fixed). The first two columns indicate whether attributePair$_{ij}$ and FCA-derived features are utilized, respectively.

	MAP	MRR	P@1	P@3	P@5	P@15
1A	76.1	92.2	92.2	82.3	76.4	71.6
1B	71.2	93.4	93.4	78.5	70.9	65.7
1C	81.0	95.9	95.9	87.2	81.7	76.4
2A	72.6	89.6	89.6	81.0	75.3	64.1
2B	95.4	98.8	98.8	97.3	96.0	93.7

Table 7: Test results of an oracle system which uses candidate filtering.

3.3 Post-evaluation analysis

After the evaluation closed, we conducted ablation experiments the results of which are included in Table 6. In these experiments, we investigated the contribution of the features derived from sparse attribute pairs and FCA. These ablation experiments corroborate the importance of features derived from sparse attribute pairs and reveal that turning off FCA-based features does not hurt performance at all. For this reason – even though our official shared task submission included FCA-related features – we no longer employed them in our post-evaluation experiments.

Table 5 includes the detailed behavior of our model on subtask 1A with respect three distinct factors, that is

1. the number of basis vectors employed during sparse coding ($k \in \{200, 300, 1000\}$),
2. the number of negative training samples per positive sample ($ns \in \{50, all\}$),
3. candidate filtering being turned on/off.

In our original submission we generated 50 negative samples (ns) generated per query q during training. In our post evaluation experiments we investigated the effects of generating more negative samples, i.e. we regarded all the valid hypernyms over the training set – not being a proper hypernym for q – as h' upon the creation of the (q, h') negative training instances. This latter strategy is referenced as $ns = all$ in Table 5.

In our official submission we regarded only those hypernyms as potential candidates to rank during test time which occurred at least once as a correct hypernym in the training data. We call this strategy as candidate filtering. Historically, we applied this restriction to speed up the FCA algorithm because this way the size of the concept lattice could be made smaller. As there are valid hypernyms on the test set which never occurred in the training data, our official submission would not be able to obtain a perfect score even in theory. Table 7 contains the best possible metrics on the test set that we could achieve when candidate filtering is applied. In our post evaluation experiments we also investigated the effects of turning this kind of filtering step off. As Table 5 illustrates, however, our scores degrade after turning candidate filtering off.

Our post evaluation experiments in Table 5 sug-

	MAP	MRR	P@1	P@3	P@10	P@15
1A	13.3	28.1	21.3	13.8	12.6	12.3
1A	19.8	36.1	29.7	21.1	19.0	18.3
1B	12.5	24.2	14.5	13.4	12.5	12.0
1B	12.1	25.1	17.6	12.9	11.7	11.2
1C	21.8	43.8	33.7	22.9	21.4	19.9
1C	20.0	28.3	21.4	20.9	21.0	19.4
2A	21.9	39.5	34.2	25.5	22.6	18.5
2A	34.0	54.6	49.2	40.1	36.8	27.1
2B	31.5	43.6	29.8	30.3	30.3	31.5
2B	41.0	60.9	48.2	44.9	41.3	38.0

Table 8: Post evaluation results for the different subtasks using $k = 1000, ns = 50$ and hypernym candidate filtering. Upper: our system, lower: subtask winner.

gest that it is advantageous to apply sparse representation of more expressive power (i.e. a higher number of basis vectors). Generating more negative samples also provides some additional performance boost. These previous observations hold irrespective whether candidate filtering is employed or not, however, their effects are more pronounced when hypernym candidates are not filtered.

Finally, we report our post-evaluation results for all the subtasks and compare them to the official scores of the best performing systems in Table 8. It can be seen from these enhanced results for category "all" (concepts and entities mixed) that we would win (1B) Italian and (1C) Spanish. Our post-evaluation system – which only differs from our participating system that it fixes the calculation of a features, does not rely on FCA-based features and uses $k = 1000$ – would also place third in the rest of the subtasks.

4 Conclusion

In this paper we experimented with the integration of sparse word representations into the task of hypernymy discovery. We strived to utilize sparse word representations in two ways, i.e. via building concept lattices using formal concept analysis and modeling the hypernymy relation with the help of interaction terms. While our former approach for deriving formal concepts from sparse word representations was not successful, the interaction terms derived from sparse word representations proved to be highly beneficial.

Acknowledgements

We would like to thank András Kornai for useful comments on negative sampling. This research was 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.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear algebraic structure of word senses, with applications to polysemy. *arXiv:1601.03764v1*.

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

Gábor Berend. 2018. Towards cross-lingual utilization of sparse word representations. In *MSZNY2018, XVI. Magyar Számítógpes Nyelvészeti Konferencia*.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. 2018. SemEval-2018 Task 9: Hypernym Discovery. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *Journal Artificial Intelligence Research (JAIR)*, 24:305–339.

Dominik Endres, Peter Földiák, and Uta Priss. 2010. An Application of Formal Concept Analysis to Semantic Neural Decoding. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):233–248. Reviewed.

Manaal Faruqui, Jesse Dodge, Sujay Jauhar, Chris Dyer, Ed Hovy, and Noah Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL 2015*. Best Student Paper Award.

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.

Bernhard Ganter and Rudolf Wille. 2012. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR 2013)*.

Bruno A Olshausen and David J Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ron Rubinstein, Michael Zibulevsky, and Michael Elad. 2008. Efficient implementation of the k-svd algorithm and the batch-omp method. *Department of Computer Science, Technion, Israel, Tech. Rep.*

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. *AAAI*.

Dani Yogatama, Manaal Faruqui, Chris Dyer, , and Noah A. Smith. 2015. Learning word representations with hierarchical sparse coding. In *ICML*. Previous version in NIPS Deep Learning and Representation Learning Workshop 2014.

Peng Zhao, Gulherme Rocha, and Bin Yu. 2009. The composite and absolute penalties for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497.