Document Representations using Fine-Grained Topics

Justin Payan and Andrew McCallum
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01002
{jpayan,mccallum}@cs.umass.edu

Abstract

Pre-trained language models, which provide dense embedded representations of words in context, achieve state-of-the-art results on countless NLP tasks. While highly effective, these dense representations lack some of the advantages of their sparse counterparts, including simple inverted indexing and interpretability. We propose a method for building sparse topical features for documents using a hierarchical clustering over contextualized token representations in a corpus. We show that these sparse features are effective representations for document classification, outperforming other sparse representations by 12 points on one dataset, while only performing slightly worse than dense representations.

1 Introduction

Dense representations of words, sentences, and documents have achieved phenomenal success with state-of-the-art results in most if not all natural language tasks [8, 16, 19, 14, 18, 20] by pre-training models on massive text corpora. Recent methods also incorporate context to capture multiple word senses and generalize to word usages not seen at training time.

Sparse features have several desirable properties not found in their dense counterparts. Sparse features are easily indexed via inverted indices to facilitate efficient lookup of nearest neighbors. They often provide interpretable topic-like representations, which can be used to explain why documents are related or why a given classification decision is made. Sparse features can also be used to efficiently and dynamically short-cut classification decisions making them more efficient [27].

Often these sparse features derive from global features of documents or hand-crafted linguistic constructs, rather than leveraging the fine-grained nuance contained in dense, contextualized representations. A natural means of deriving global structure from existing representations is clustering, which has often been used to discover meaningful sparse features. Topic models implicitly perform clustering of the underlying bag-of-words representations, and Brown clusters quite notably cluster words in similar contexts using hierarchical agglomerative clustering. However, these methods are limited in their ability to incorporate polysemy and handle new word types/senses.

In this paper, we present a clustering-based approach for generating sparse features for words and documents using a combination of pre-trained language model word representations and hierarchical clustering. The hierarchical representation allows our approach to model multiple granularities of clusters, more fine-grained than standard topic modeling based approaches. We use methods for online and incremental hierarchical clustering to help support discovery of new topics as they emerge in data that arrives in an online fashion. We use *contextualized* word representations to help mitigate challenges of polysemy. We perform experiments on document classification tasks and compare to existing modern and classic sparse representation-based approaches. We find that our approach outperforms these state-of-the-art sparse representations by 1 point on sentiment classification and over 10 points on multiclass classification, while performing not much worse than state-of-the-art dense methods.

Sets & Partitions Workshop at 33rd Conference on Neural Information Processing Systems (NeurIPS 2019).

Related Work Sparse features have a long history in natural language processing with models using bag-of-words [23], Brown clusters [4], topic-based features [3, 2], and parse features applied to a wide variety of tasks such as document classification [17], named entity-recognition [21], semantic parsing [30], relation extraction [7, 22], and others. Sparse representations of data items (such as words/documents) are frequently discovered using methods like Latent Dirichlet Allocation [3], non-negative matrix factorization, dictionary learning [10], representation learning [25, 6], and recently using neural variational topic models [26, 5, 12]. Bayesian non-parameteric methods such as the Indian Buffet Process [9] discover infinite dimensional binary sparse vectors. This is closely related to our approach which grows the hierarchical clustering in the presence of more data. Recent work in information retrieval [31] directly learns a sparse encoding of documents such that related documents share sparse dimensions. Some recent work leverages sparsity in semantic space to improve dense representations, such as P-SIF [11] or Word Mover's Embedding [28].

2 Discovering Sparse Representations with Hierarchical Clustering

Our approach discovers topic clusters of words in context by performing a hierarchical clustering of embedded representations from a pre-trained language model. The position of a word in the hierarchical clustering defines a set of topics to which the word belongs. Topic sets for all words in one document combine to form the document topic distribution.

2.1 Dense Representations from Pre-trained Language Models

Given a corpus of documents $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, we use a pre-trained language model, in particular, BERT [8], to generate contextualized representations for each token in each document. Tokens are determined by the pre-trained language model's tokenizer and consist either of words or subwords/wordpieces [29]. The embedding of each word is based on both its forward and backward context, as the model has been trained to perform a cloze task. For a document d_i, we use D_i to represent the set of its word representations and use w_{ij} to represent the j^{th} token's embedding. After obtaining the embeddings for each word, we remove stopwords, sub-word pieces, and `[CLS]` and `[SEP]` representations from each D_i and use representations from the final layer of BERT.

2.2 Incremental Hierarchical Clustering

PERCH is an incremental hierarchical clustering algorithm [13]. PERCH works by observing one data point at a time and updating the hierarchical clustering with each observation. The algorithm endows each tree node n in the hierarchical clustering with a bounding box covering its descendant data points. Each new data point, x, is efficiently routed to its nearest neighbor, n, using the bounding boxes in a manner similar to an R-tree. The new point is added as the sibling of its nearest neighbor, creating a new internal node as the parent of these two leaves. The algorithm then performs local re-arrangements, *rotations*, which swap the newly added point x with its aunt a in the tree under certain conditions. The swap is applied if the maximum distance between n, the sibling of x, and the aunt a is less than the minimum distance between n and x. The minimum distances between internal nodes is measured as the minimum distance between any descendant leaves of the nodes and is estimated using the bounding boxes. Maximum distance is defined similarly. See Figure 1.

Practical Implementation Details While PERCH is highly scalable, it can only be moderately parallelized. To scale to corpora with several million words or more, we, in a manner akin to finding coresets [1], first group words into small clusters of related words in subsets of the corpora in parallel using Mini-Batch K-means [24]. This is highly parallelizable and can effectively summarize the several million or more word corpus into a collection on the order of a hundred thousand vectors. A preliminary analysis shows the K-means cluster centers consist of only a handful of word types, often from the same document.

2.3 Sparse Word and Document Representations

We use the placement in the hierarchical clustering of each word in a document to determine the sparse representation of the document. For a given word w_{ij} in document D_i we represent this as a function of its ancestors in the tree. In particular, given a hierarchical clustering $\mathcal{T}$, we represent a word w_{ij} as the $|\mathcal{T}|$-dimensional sparse vector containing ones for its k closest ancestors (its

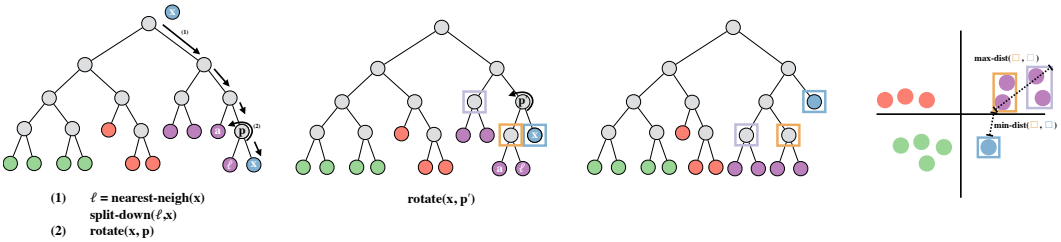

Figure 1: PERCH incremental construction and rotations. The datapoint x is first added as the sibling of its nearest neighbor ℓ. Then two recursive rotations are applied. The resulting structure keeps the points in the purple colored cluster in a subtree separate from the newly added blue point.

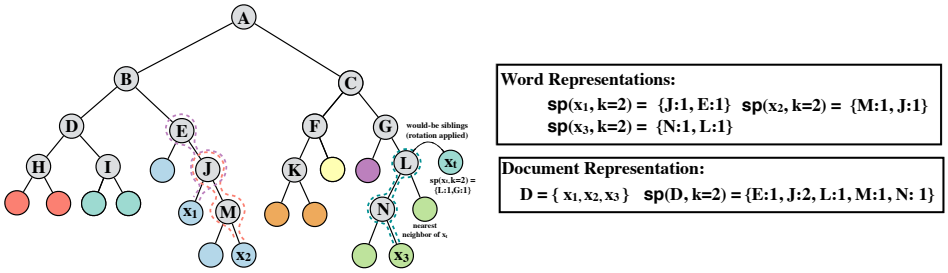

Figure 2: Example encodings in hierarchical clustering. The figure shows the sparse codes corresponding to three leaves of the tree structure x_1, x_2, and x_3. It also shows how a point x_t can be encoded but *not* added to the tree structure. The nearest neighbor of x_t is labeled and in the PERCH construction process x_t would be rotated to keep the light green points in a separate subtree. The sibling of x_t is L, and x_t is encoded by the ancestors of L.

parent, parent's parent, and so on), where $|\mathcal{T}|$ is the number of internal nodes in the hierarchical clustering. We denote this as $\mathsf{sp}(w_{ij}, k)$. We represent each document as the sum of the sparse word representations. That is, $\mathsf{sp}(D_i, k) = \sum_{w_{ij} \in D_i} \mathsf{sp}(w_{ij}, k)$. See Figure 2 for a visual representation.

In some cases, we might want to provide an encoding of a document without adding its words into the tree structure. This could be needed for time or space efficiency or to achieve parallelism in encoding. To achieve this, for each word in a document, we determine which existing tree node *would* be the sibling of that word were it added in PERCH (i.e., after rotations are applied). The construction algorithm of PERCH would create a new parent node for the newly added node and the found sibling. The creation of this node would effectively create a new dimension in the encoding vector space.

PERCH has particularly desirable properties for online topic discovery. Rotations in PERCH never remove a previously existing ancestor from any given leaf's set of ancestors, meaning that existing dimensions of previously-encoded documents' sparse codes do not need to be edited when new documents are added. New dimensions may still need to be added to existing documents. Both properties are apparent in Figure 1, where leaves a and l never rise above the level of the orange node (and hence maintain all their existing ancestors). However, a and l do gain p as an ancestor.

3 Experiments

We evaluate the quality of our sparse representations by using our unsupervised sparse representations as input to a supervised classifier. We also provide a qualitative analysis of the topics discovered by our method.

We evaluate on two datasets: **Twenty News Groups**, text documents labeled with one of twenty news groups categories; and **IMDB review sentiment classification**, a binary sentiment analysis dataset containing the text of movie reviews [15].

To evaluate the quality of document representations, we use the document representations to form feature vectors for training a supervised classifier for each dataset. This follows the experimental setup of Card et al. [5] and Gupta et al. [11]. We train a logistic regression classifier as in Card et

	Sparse						Dense
	Our Approach (k=1)	Our Approach (k=5)	Scholar (labels) [5]	Scholar (covariates) [5]	BoW [5]	SLDA [5]	P-SIF [11]
20NG	0.82	0.83	0.67	0.71	0.70	0.60	0.86
IMDB	0.87	0.88	0.86	0.87	0.87	0.64	–

Table 1: Document classification accuracy of baseline models compared with our approach. Baseline results from [5]. P-SIF did not evaluate on IMDB.

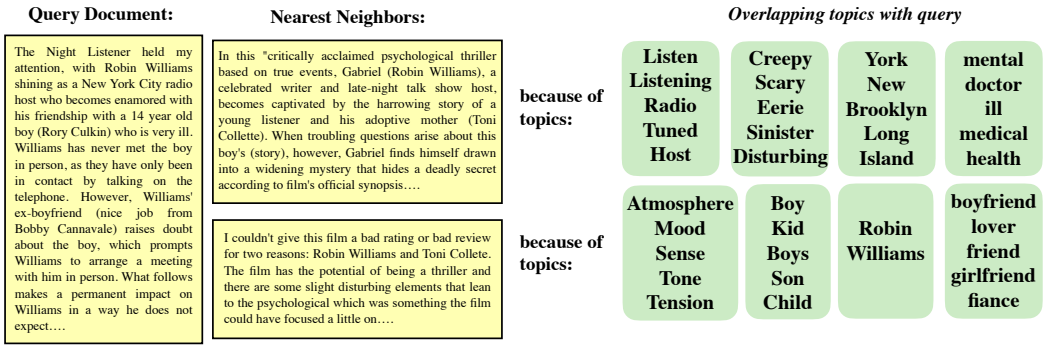

Figure 3: Example query IMDB review, with the first 2 retrieved results.

al. [5]. We split the training data into a 80%/20% train/dev split and perform a grid-search over the regularization parameters of the classifier, as well as early stopping on the dev set.

We compare our approach to a recently proposed state-of-the-art sparse representation, **Scholar**, proposed by Card et al. [5]. Scholar is a neural topic model that also supports supervision and incorporating covariates/document meta-data. We also compare our approach to the state-of-the-art dense document representation method P-SIF [11].

For our method, we use the BERT-base-uncased pre-trained BERT model without any fine-tuning. We build the hierarchical clustering on the training dataset. The encoding method that encodes a document without adding its words to the tree is used. We tested with the number of ancestors parameter k ranging from 1 to 10. The word representations from BERT are unit normed.

Table 3 provides the results for the document classification experiment. There was little variation in scores for $k > 1$, so we report scores for $k = 1$ and $k = 5$ on both datasets. We find that our method is much more accurate than the other sparse representations, while only performing slightly worse than the dense representation-based approach. We hypothesize that our improved performance comes from the quality of the pre-trained language model that drives our sparse representation. Our clustering allows the representation to encode the groups of related words in fine-grained topics. We illustrate what the discovered fine-grained topics look like by providing example documents and nearest neighbors by topics in Figure 3. For each neighbor, we show several of the shared topics.

4 Conclusion

We introduce a method for building sparse word and document representations using a hierarchical clustering of word token representations from a pre-trained language model. We show how these features can be effectively used for document classification. We also demonstrate the interpretability of these topics by inspecting which topics are present in pairs of nearest neighbor documents. In future work, we intend to demonstrate the model's ability to adapt online to topic introduction and drift, as well as develop training methods to jointly update the clustering structure, sparse codes, and word representations.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. We thank Nicholas Monath for ideas and guidance. This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by the Chan Zuckerberg Initiative under the project "Scientific Knowledge Base Construction, and in part by the National Science Foundation under Grant No. NSF-1763618. The work reported here was performed in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

[2] Somnath Banerjee. Improving text classification accuracy using topic modeling over an additional corpus. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 867–868. ACM, 2008.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[5] Dallas Card, Chenhao Tan, and Noah A. Smith. Neural models for documents with metadata. In *Proceedings of ACL*, 2018.

[6] Suthee Chaidaroon and Yi Fang. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84. ACM, 2017.

[7] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 423. Association for Computational Linguistics, 2004.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[9] Zoubin Ghahramani and Thomas L Griffiths. Infinite latent feature models and the indian buffet process. In *Advances in neural information processing systems*, pages 475–482, 2006.

[10] Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. Document similarity for texts of varying lengths via hidden topics. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2341–2351, 2018.

[11] Vivek Gupta, Ankit Kumar Saw, Partha Pratim Talukdar, and Praneeth Netrapalli. Unsupervised document representation using partition word-vectors averaging, 2019.

[12] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A Smith. Variational pretraining for semi-supervised text classification. *arXiv preprint arXiv:1906.02242*, 2019.

[13] Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. An online hierarchical algorithm for extreme clustering. *arXiv preprint arXiv:1704.01858*, 2017.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[15] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[17] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

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

[19] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

[20] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[21] Lev-Arie Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 2009.

[22] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[23] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 1975.

[24] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

[25] Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Ricardo Henao, and Lawrence Carin. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2041–2050, 2018.

[26] Akash Srivastava and Charles Sutton. Autoencoding variational inference for topic models. In *International Conference on Learning Representations (ICLR)*, 2017.

[27] Emma Strubell, Luke Vilnis, Kate Silverstein, and Andrew McCallum. Learning dynamic feature selection for fast sequential prediction. 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 146–155, 2015.

[28] Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock. Word mover's embedding: From word2vec to document embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4524–4534, 2018.

[29] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[30] Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94, 2004.

[31] Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 497–506. ACM, 2018.