A Compressed Sensing View of Unsupervised Text Embeddings, Bag-of-n-Grams, and LSTMs

Sanjeev Arora
Mikhail Khodak
Nikunj Saunshi
Kiran Vodrahalli

Overview

Motivation:

- Success of modern NLP is based around *distributed representations* - low-dimensional semantic text embeddings that are used and produced by neural networks.

- Deep networks work well in practice but are not yet dominant in all NLP tasks and are largely uninterpretable

Goal:

Reason formally about distributed representations for text:

- What information do they encode?
- How will they perform on downstream tasks?

Contributions

Theoretical Results

- We prove that LSTMs can compute compressed representations of simple (but very effective) sparse feature representations (e.g. Bag-of-Words) that are *approximately* as powerful for linear document classification.

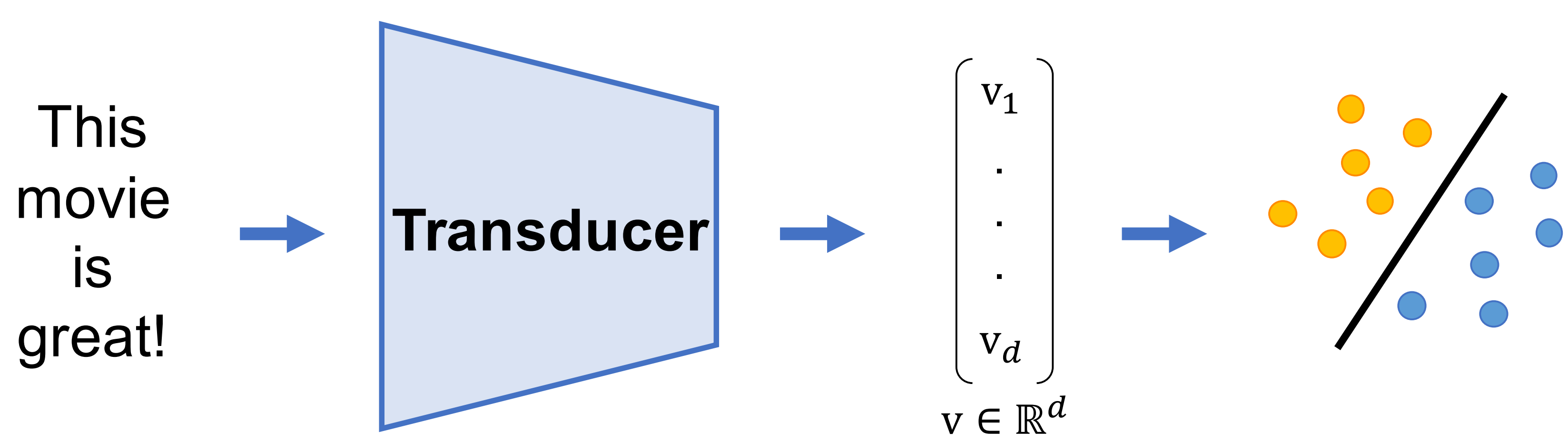
Empirical finding

- We also observe empirically that word embeddings provide a surprisingly effective design matrix for sparse recovery of Bag-of-Words.

Setting

- Assume a distribution **D** of documents, each a sequence of at-most **T** words $\mathbf{w_1, \ldots, w_T}$ drawn from a vocabulary of size **V**.

- We are interested in fixed-dimensional document representations over which we can learn a binary linear classifier.



Sparse Representation: Bag-of-n-Grams (BonG)

- Bag-of-Words: represent each document by a vector counting the number of times each word appears.

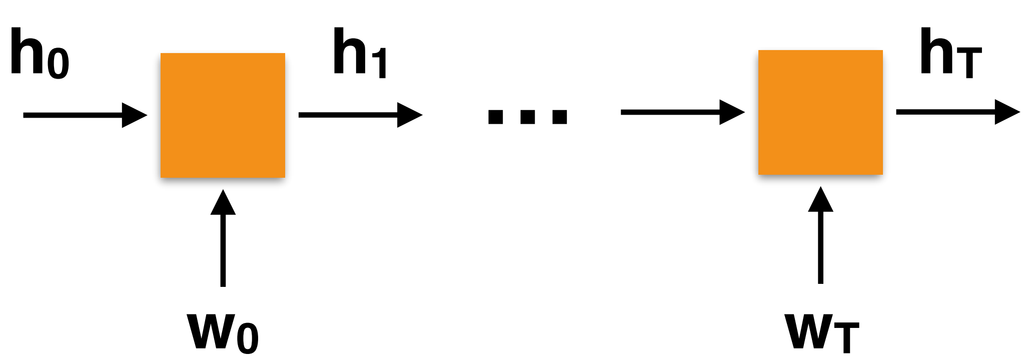
- Bag-of-n-Grams: represent each document by a vector counting the number of times each unigram, bigram, …, n-gram appears.
  - Surprisingly effective (Wang & Manning 2012).

Distributed Representation: Linear Scheme

- Assign a real-vector $\mathbf{v_w}$ to every word $\mathbf{w}$. Take a sum of the vectors of word in a document.
  - Empirically shown to be effective on some tasks (Wieting et al. 2016, Arora et al. 2017)
  - Can be viewed as a linear compression $\mathbf{Ax}$ of the BoW vector $\mathbf{x}$, where the columns of $\mathbf{A}$ are the vectors $\mathbf{v_w}$

Distributed Representation: LSTM

- Assign a real-vector $\mathbf{v_w}$ to every word $\mathbf{w}$. An LSTM takes a sequence of words $(\mathbf{w_1}, \ldots, \mathbf{w_T})$ as input and computes a hidden state vector $\mathbf{h_t}$ at each word in document as follows



$$h_t = F(v_{w_t}, h_{t-1})$$

$$f(v_{w_t}, h_{t-1}) \circ h_{t-1} + i(v_{w_t}, h_{t-1}) \circ g(v_{w_t}, h_{t-1}))$$

- Represent the document as the last state $\mathbf{h_T}$.

- Use (un)supervised training to learn the LSTM parameters.

Related Work on BonG Compression

- Compressed representation that can recover BonG vector
  - Plate (1995): represent objects (words) using low-dimensional random vectors, compose objects (n-grams) using circular convolution, and represent collections of items (documents) using summation.
  - Paskov et al. (2013): use a LZ77-inspired approach to reduce the number of features; good classification performance but still quite high-dimensional.
- None of them analyze performance on downstream tasks.

Main Theorem

Theorem [AKSV'18]: Let w_0 be the optimal linear classifier for BonGs for some convex Lipschitz loss ℓ. Then we can initialize a $\mathcal{O}(nd)$-memory LSTM and learn a linear classifier $\hat{w}$ so that with probability $1-\delta$

$$\ell(\hat{w}) \leq \ell(w_0) + \mathcal{O}\left(\|w_0\|_2 \sqrt{\varepsilon + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

for $d = \tilde{\Omega}\left(\frac{T}{\varepsilon^2}\log\frac{nV}{\delta}\right)$. Here T is the maximum document length, V is the vocabulary size, and m is the number of samples.

Proof Outline

- Design an RIP matrix **A** such that a *low-memory* LSTM can compute a document representation **Ax**, where **x** is a BonG vector.

- Show that learning is possible under compression: a linear classifier learned over $\{\mathbf{Ax_i}\}$ is almost as good as a linear classifier learned over $\{\mathbf{x_i}\}$ if the vectors $\mathbf{x_i}$ are sparse and **A** satisfies an RIP condition.

Restricted Isometry Property

A is (k, ϵ)-RIP if $(1-\epsilon)\|x\|_2 \leq \|Ax\|_2 \leq (1+\epsilon)\|x\|_2$ for all k-sparse x

Assumptions

- n-grams are order-invariant ((a,b) ~ (b,a))
  - reasonable - performance is about the same
- no word occurs in any n-gram more than once (no (a,a), (a,b,a))
  - violated in real documents, but can be removed by a preprocessing step

Proof Outline

- Design an RIP matrix **A** such that a low-memory LSTM can compute a document representation **Ax**, where **x** is a BonG vector.

- Show that learning is possible under compression: a linear classifier learned over $\{\mathbf{Ax}_i\}$ is almost as good as a linear classifier learned over $\{\mathbf{x}_i\}$ if the vectors $\mathbf{x}_i$ are sparse and **A** satisfies an RIP condition.

Document Representation

Words: For every word w sample i.i.d. $v_w \sim \frac{1}{\sqrt{d}}\{\pm 1\}^d$

n-gram: For $g = w_1, \ldots, w_n$, use element wise product of word vectors

$$v_g = v_{w_1} \circ \cdots \circ v_{w_n}$$

Document: Sum of p-gram embeddings for all $p \leq n$

$$v_D = \sum_{p \leq n} \sum_{g \in p-\text{gram}} v_g$$

Linear Compression

$$v_D = Ax_{BonG}$$

where the columns of A
are the n-gram embeddings

Compositionality

v_D can be computed using

a low-memory LSTM

Randomness

A is (T, ϵ)-RIP for

$$d = \tilde{\mathcal{O}}\left(\frac{T}{\epsilon^2}\right)$$

Proof Outline

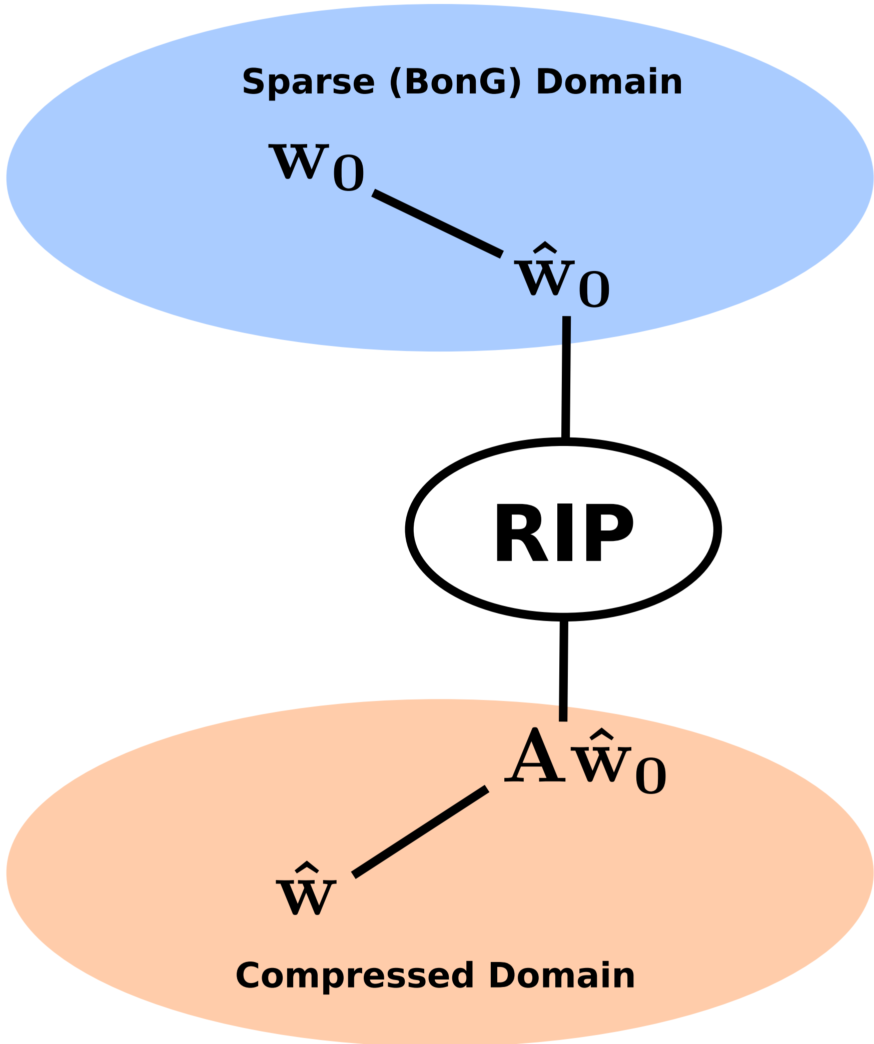
- Design an RIP matrix **A** such that a low-memory LSTM can compute a document representation **Ax**, where **x** is a BonG vector.

- Show that learning is possible under compression: a linear classifier learned over $\{\mathbf{Ax}_i\}$ is almost as good as a linear classifier learned over $\{\mathbf{x}_i\}$ if the vectors $\mathbf{x}_i$ are sparse and **A** satisfies an RIP condition.

Compressed Learning (Calderbank et al. 2009)

We examine four different classifiers:

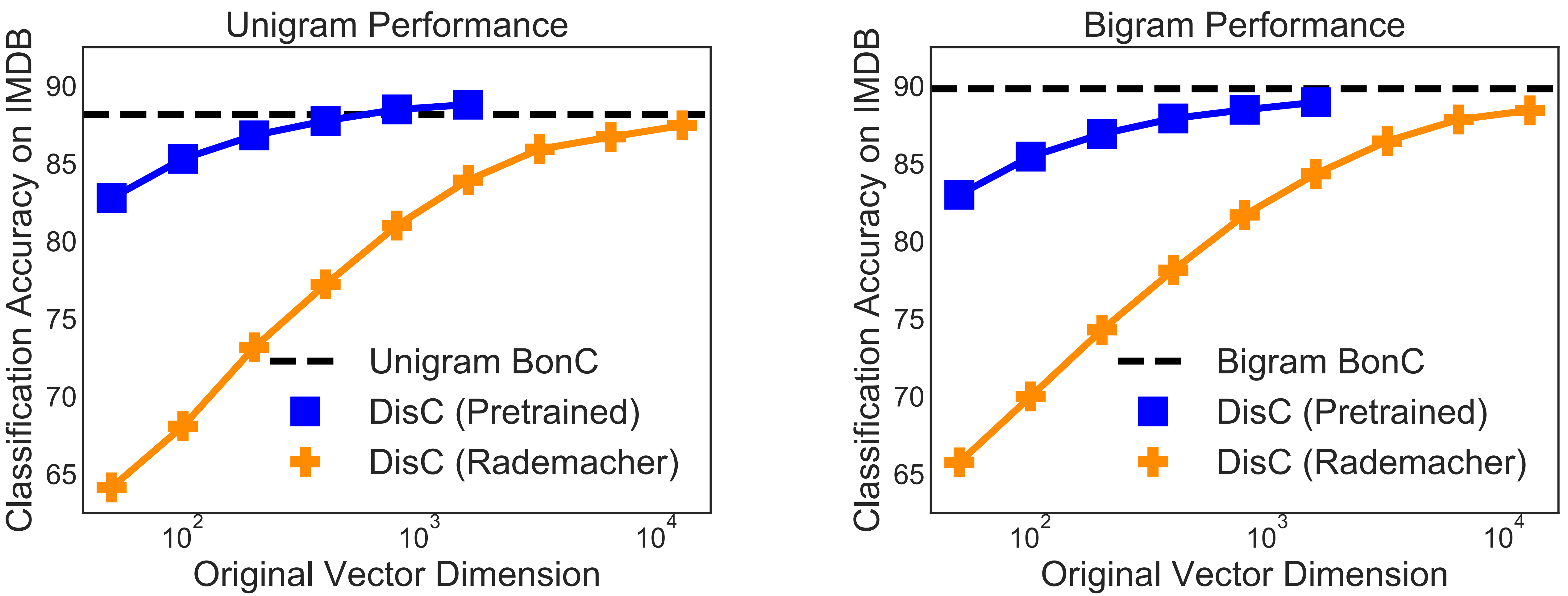
1. the optimal sparse classifier $\mathbf{w_0}$
2. the sparse classifier $\hat{\mathbf{w}}_\mathbf{0}$ minimizing the (regularized) loss over $\{(x_i, y_i)\}_{i=1}^m$
3. the dense classifier $\mathbf{A}\hat{\mathbf{w}}_\mathbf{0}$
4. the classifier $\hat{\mathbf{w}}$ minimizing the (regularized) loss over $\{(Ax_i, y_i)\}_{i=1}^m$



Bounding $\ell(\hat{w}_0)$ in terms of $\ell(w_0)$ and $\ell(\hat{w})$ in terms of $\ell(A\hat{w}_0)$ can be done using standard techniques. **We need the RIP condition on A to bound $\ell(A\hat{w}_0)$ in terms of $\ell(\hat{w}_0)$.**

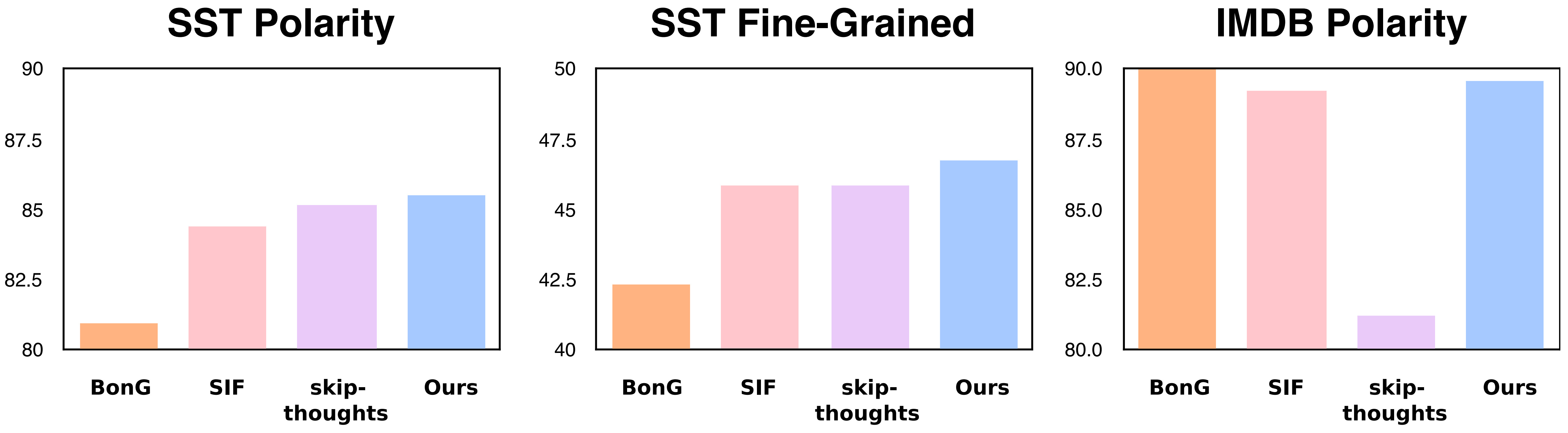
Classification Performance

$$\ell(\hat{w}) \leq \ell(w_0) + \mathcal{O}\left(\|w_0\|_2 \sqrt{\varepsilon + \frac{1}{m}\log\frac{1}{\delta}}\right) \qquad d = \tilde{\mathcal{O}}(\tfrac{T}{\epsilon^2})$$



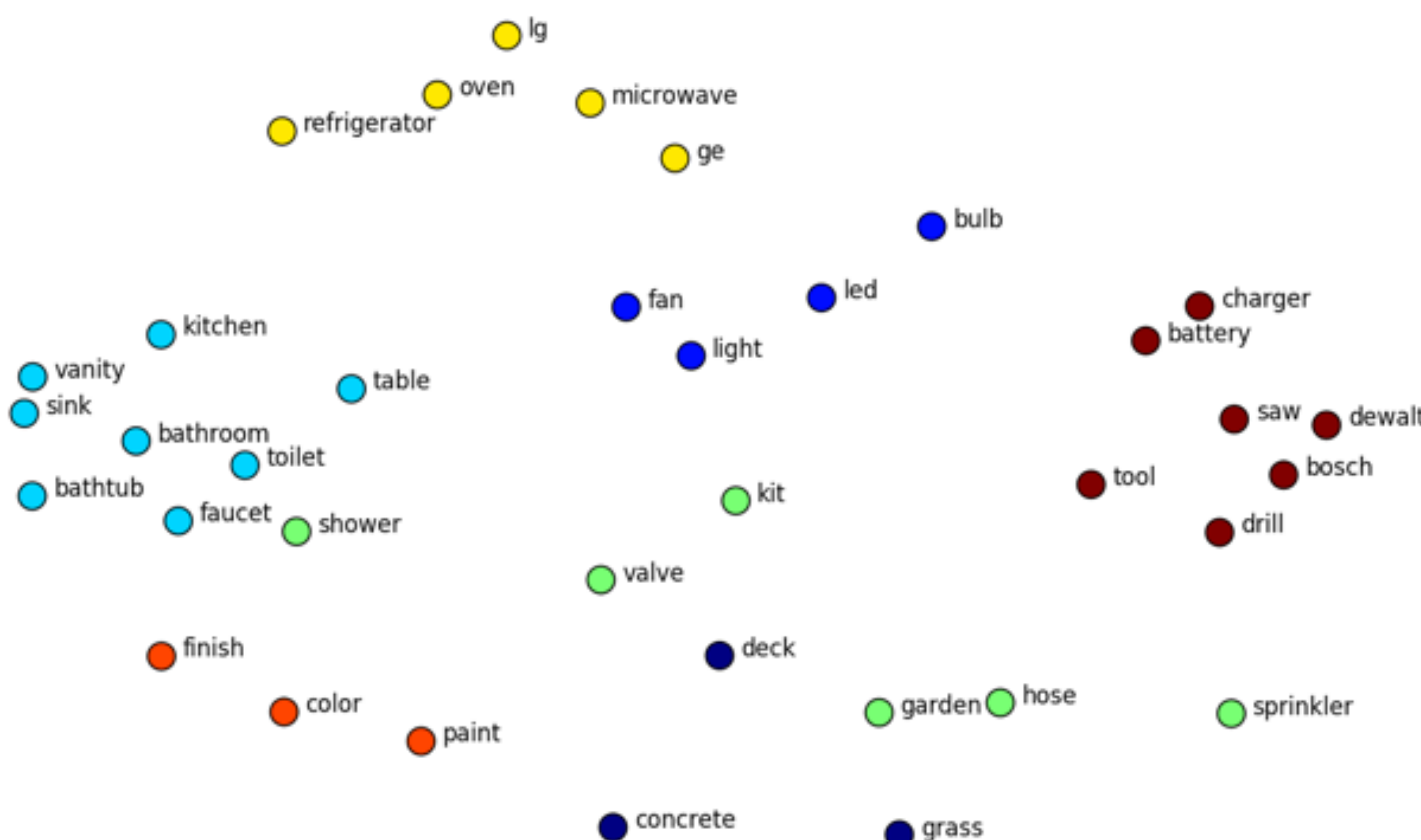
Classification Performance

- Our method is simple, compositional, and compares well against both Bag-of-n-Grams and deep LSTM representations.



Word Embeddings

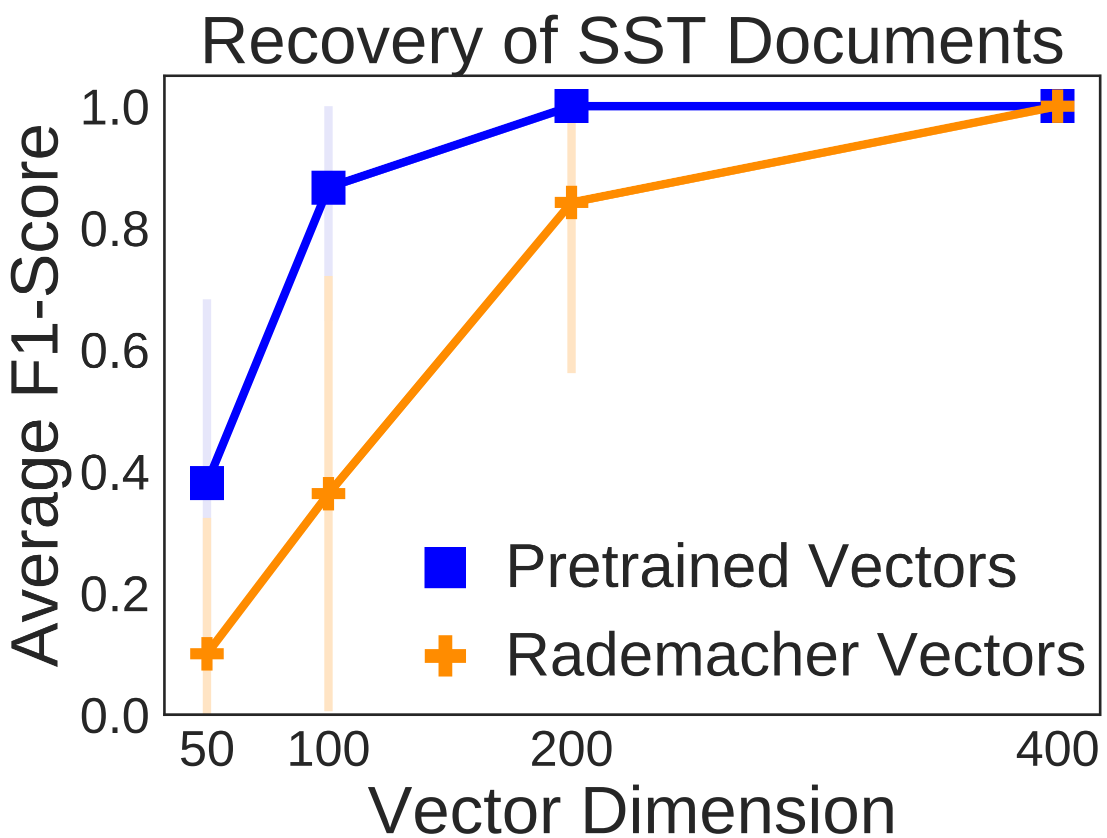
- Guarantees for compressed learning assume words represented by Rademacher random vectors.
- In practice pretrained embeddings capturing the ‘meaning’ of words are used instead.
- These vectors are trained so that similar words are closer together and thus *cannot* satisfy RIP. How can we understand their better performance?

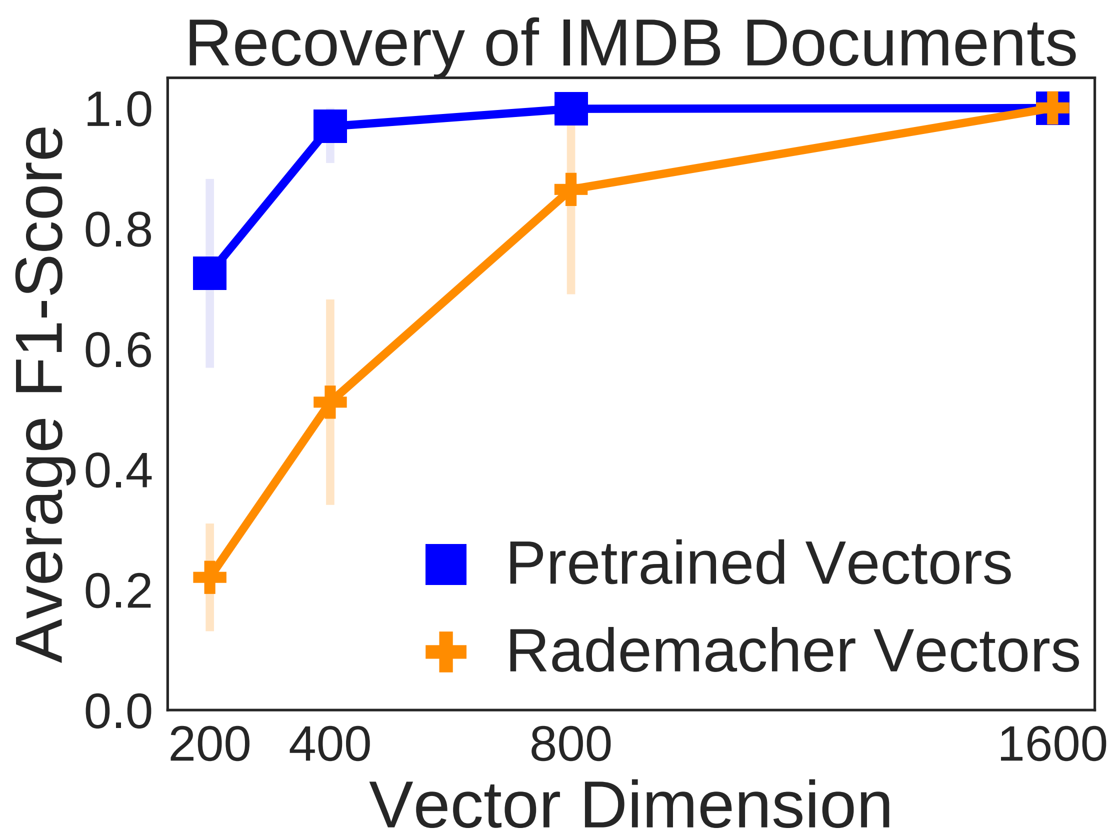


(Merentitis et al. 2016)

A Sparse Recovery Experiment

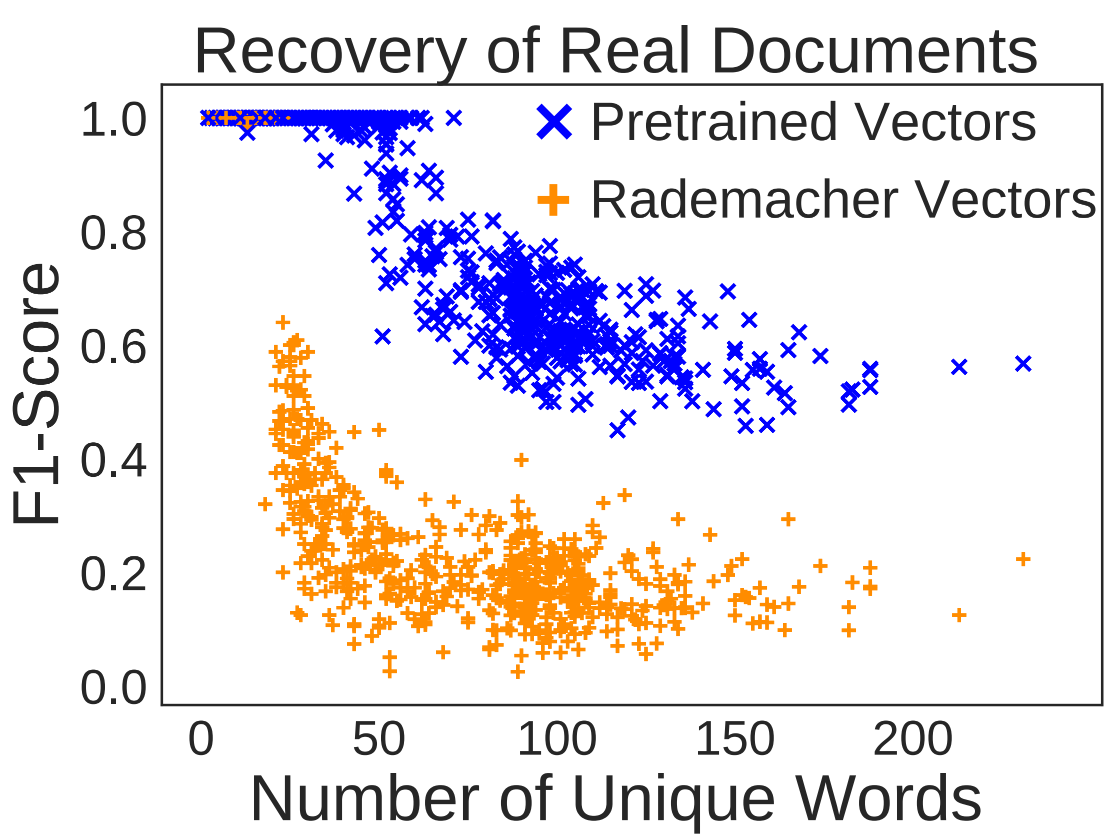
- What do word embedding-based document representations encode?
  - Compress a BoW vector **x**: **b = Ax**
  - Recover **x** using Basis Pursuit (BP): $\mathbf{min\ |x|_1\ s.t.\ Ax = b}$
  - Note: RIP provides exact recovery guarantees for BP.

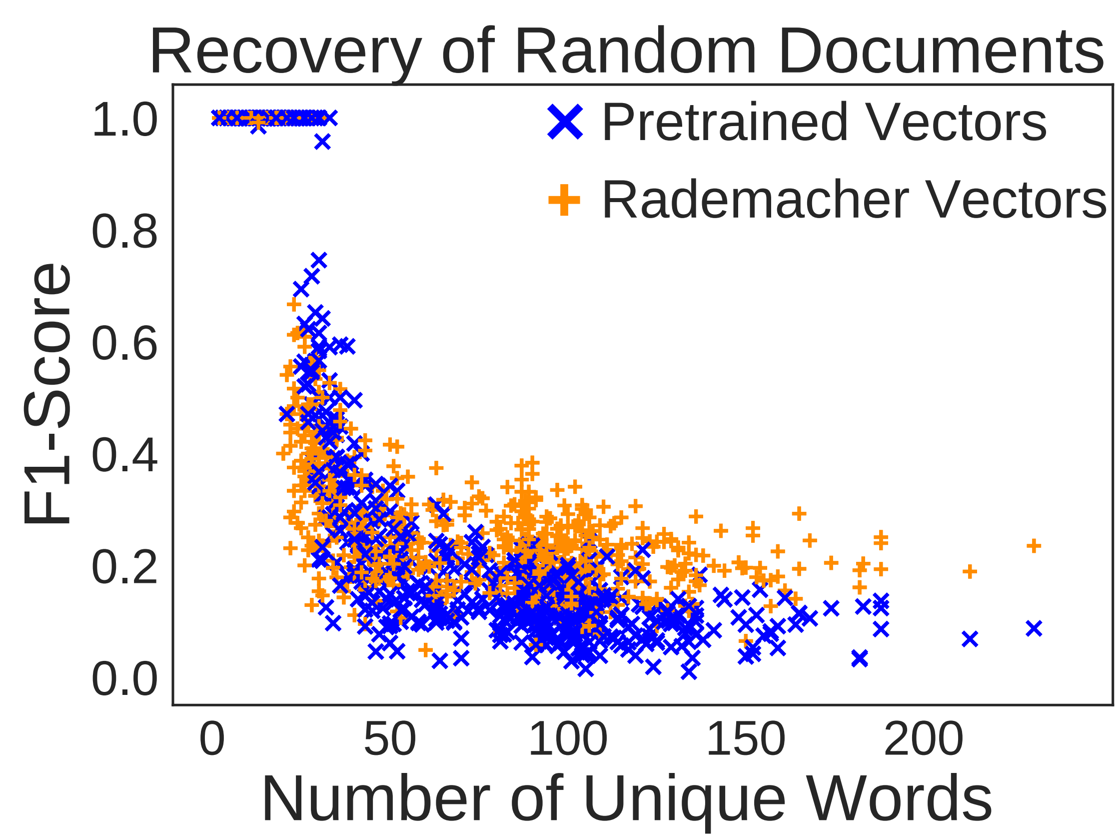




Why Are Embeddings Good for Compressed Sensing?

- RIP is a very strong condition - sufficient but not necessary
- Word embeddings only perform well when the compressed signal is a BoW vector; for random sparse vectors they perform poorly:





Recovery Properties

Restricted Isometry Property (RIP):

- guarantees recovery for all sparse signals
- **Too Strong**: does not use signal structure

Nullspace Property (NSP):

- guarantees recovery for all sparse signals **with a given support**
- do not know how to check efficiently

Nonnegative Recovery

BoW signals are nonnegative, so we can solve BP+:

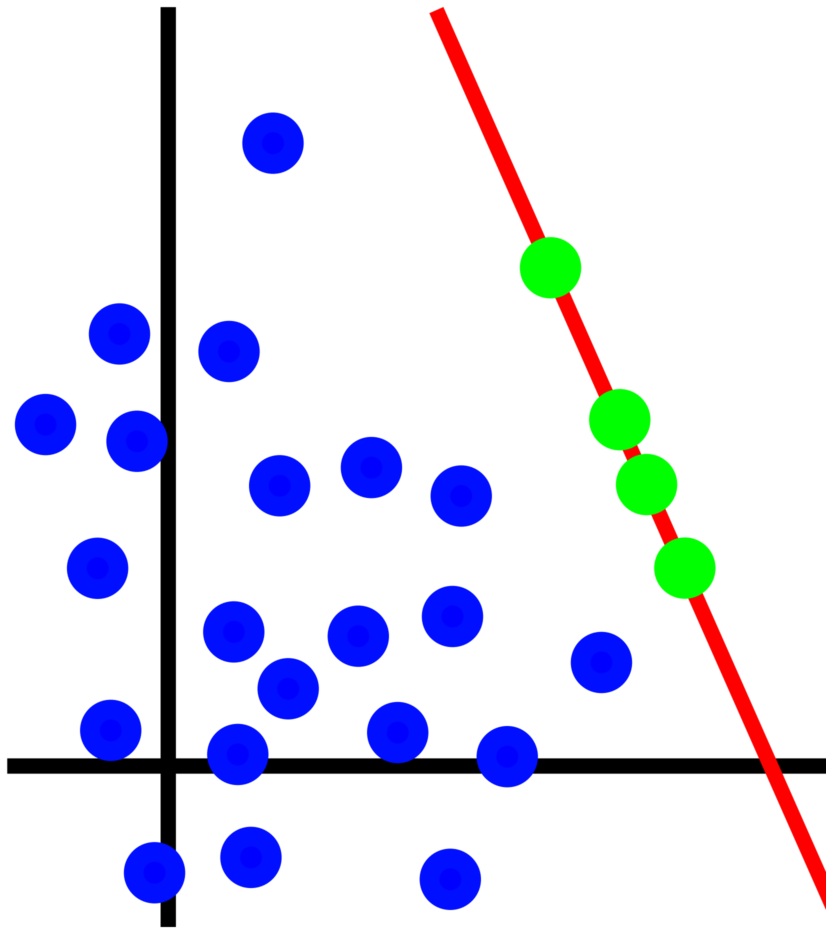
$$\min \|x\|_1 \text{ s.t. } Ax = b,\ x \geq 0$$

Donoho & Tanner (2005) (**Polytope Condition**):

BP+ recovers all x with supp(x)=S from Ax iff the columns of A indexed by S form a face of conv(A).

A Verifiable Sparse Recovery Condition

We say that a matrix A and index set S satisfy the **Supporting Hyperplane Property (SHP)** if there exists a hyperplane going through the columns of A indexed by S and all other columns of A are on the same side of the hyperplane as the origin.



Theorem:

BP+ recovers all x with supp(x) from Ax iff A and supp(x) satisfy SHP.

A Verifiable Sparse Recovery Condition

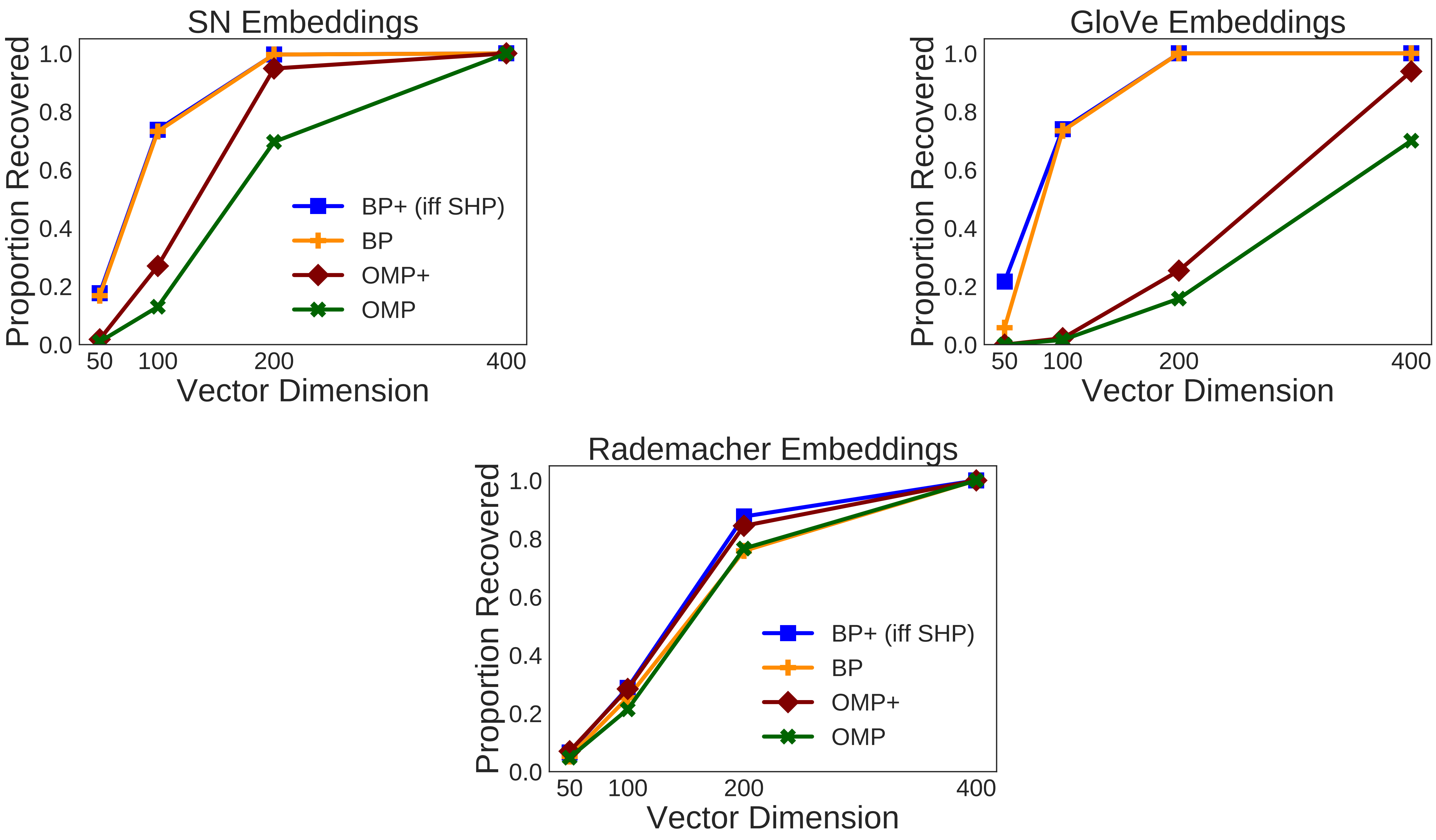
To verify SHP:

- solve the following convex program
- check if the optimal objective value is zero

$$\min_{h \in \mathbb{R}^{d+1}} \sum_{i \notin S} \max\left\{\tilde{A}_i^T h + \varepsilon, 0\right\}^p \quad \text{subject to} \quad \tilde{A}_S^T h = \mathbf{0}_{|S|}$$

$$\text{where} \quad \tilde{A} = \begin{pmatrix} A & \mathbf{0}_d \\ \mathbf{1}_N^T & 1 \end{pmatrix} \quad \text{and} \quad \varepsilon > 0,\ p \geq 1$$

Recovery vs Embedding

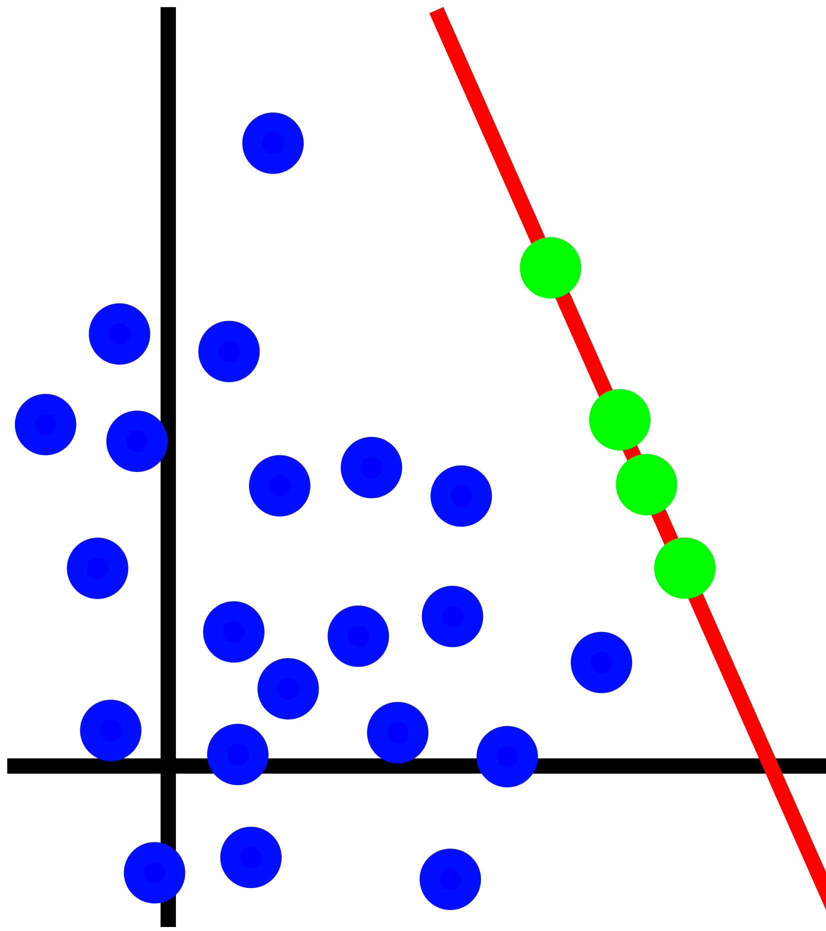


A Geometric Understanding of Recovery

Can SHP explain better recovery using word embeddings?

- Words occurring in the same document tend to have similar vectors - perhaps they are more likely to have a hyperplane separating them out.

- May be explained via a generative model of text where words are emitted based on similarity with a fixed context vector.



Future Work: Recovery vs. Classification

- Compressed learning results depend on **RIP**. Empirical results only show that word embeddings satisfy some **weaker** recovery property.

- We need an intermediate condition that:

    - provides compressed learning guarantees relative to BoW/BonG

    - guarantees recovery for certain signal distributions such as document BoW

Future Work: Applications of Recovery

- Train bigram/trigram embeddings that also recover - can reconstruct word order.

- Apply to simple encoding schemes in NLP
  - Simple approach to machine translation
  - Continuous representation for GAN training

THANK YOU