Compressed Representations of Text Documents

Papers by Paskov et al. 2013[1], 2016[3]

Presented by Misha Khodak

10 May 2017

1 Hristo S. Paskov, Robert West, John C. Mitchell, and Trevor J. Hastie. *Compressive Feature Learning*. NIPS 2013.

2 Hristo S. Paskov, John C. Mitchell, and Trevor J. Hastie. *Data Representation and Compression Using Linear-Programming Approximations*. ICLR 2016.

Compression in Unsupervised Learning

- Widespread agreement that a ‘good’ representation involves compression
  - Rate-Distortion Theory: minimize expected distortion given a constraint on the rate (number of bits needed to represent)
  - Hazan-Ma Framework: reconstruct data at least as good (probably, approximately) as a hypothesis class given a representation length constraint
  - Methods: PCA, Auto-Encoders
- How to apply this idea to text data?
  - Discrete data – harder to pose optimization problems
  - Any real-valued representation is already lossy

Overview

- Use dictionary-based compression scheme to find a succinct document representation.
- Rewrite compression scheme as an optimization problem.
- Representation similar to a common baseline approach in NLP but with dimension two orders of magnitude smaller.
- Do as well as the baseline approach on text classification tasks but with lower memory and computation costs.

Bag of N-Grams: Representation

Documents:

$$\begin{aligned} S_1 &= \text{movie was quite good} \\ S_2 &= \text{movie was not good} \end{aligned}$$

Vocabulary ($n = 2$):

$V = \{\text{movie, was, quite, good, not, (movie, was), (was, quite), (quite, good), (was, not), (not, good)}\}$

Bag of Bigrams Feature Vectors (in $\{0,1\}^{|V|}$):

$$\begin{aligned} x_1 &= \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \\ x_2 &= \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{aligned}$$

Bag of N-Grams: Drawbacks

- High-Dimensional
  - For small datasets, unigram V = 50K-100K, bigram V = 100K+, trigram V = 1M+
  - Hard-to-construct for large datasets
- Maintains no ordering information
  - ‘Movie was good but acting was terrible’ has almost the same feature vector as ‘Movie was terrible but acting was good’
- Overfitting for high values of n
  - n-grams with high values of n encode more meaning, which is desirable, but also occur more rarely, which leads to overfitting
- Poor semantic similarity
  - ‘Movie was quite good’ has zero inner product with ‘Film is pretty decent’

Some hope of fixing the first three issues using compression

Lempel-Ziv Compression: Representation

LZ77 Algorithm for Characters

- concatenate all documents
- scan from left to right until longest previously seen matching substring
- output a pointer to that previous instance and continue

We use words rather than characters:

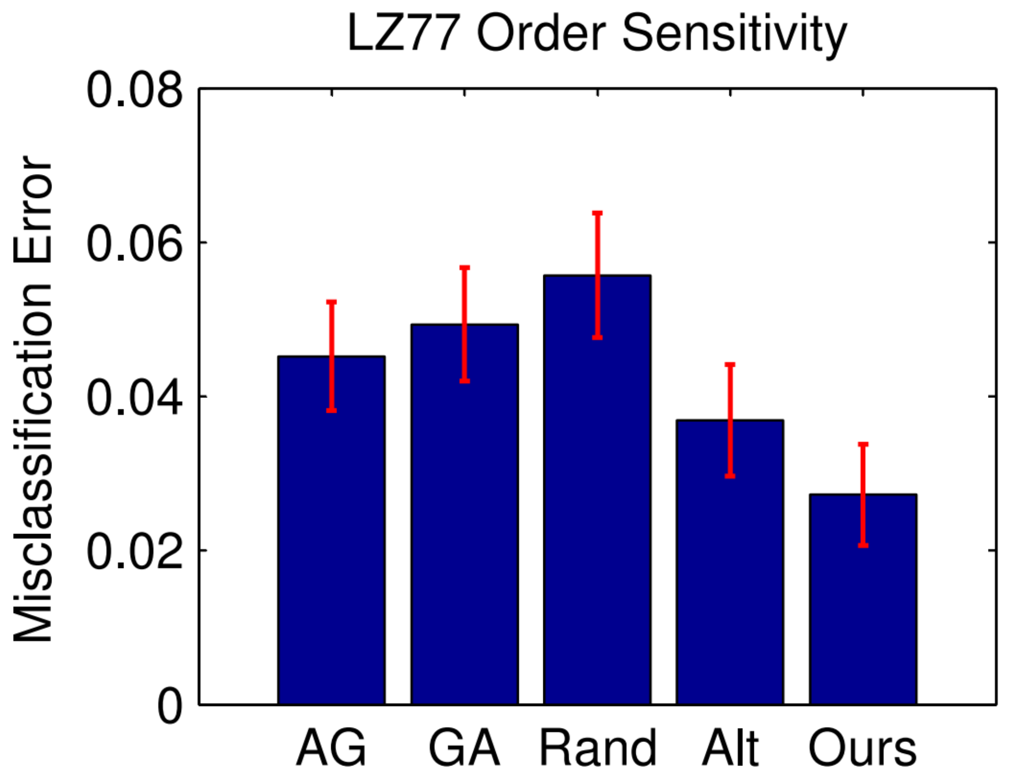
$$\begin{pmatrix} S_1 & S_2 \end{pmatrix} = \text{movie was quite good movie was not good}$$

$$V = \{\text{movie, was, quite, good, not, (movie, was)}\}$$

$$x_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$
$$x_2 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

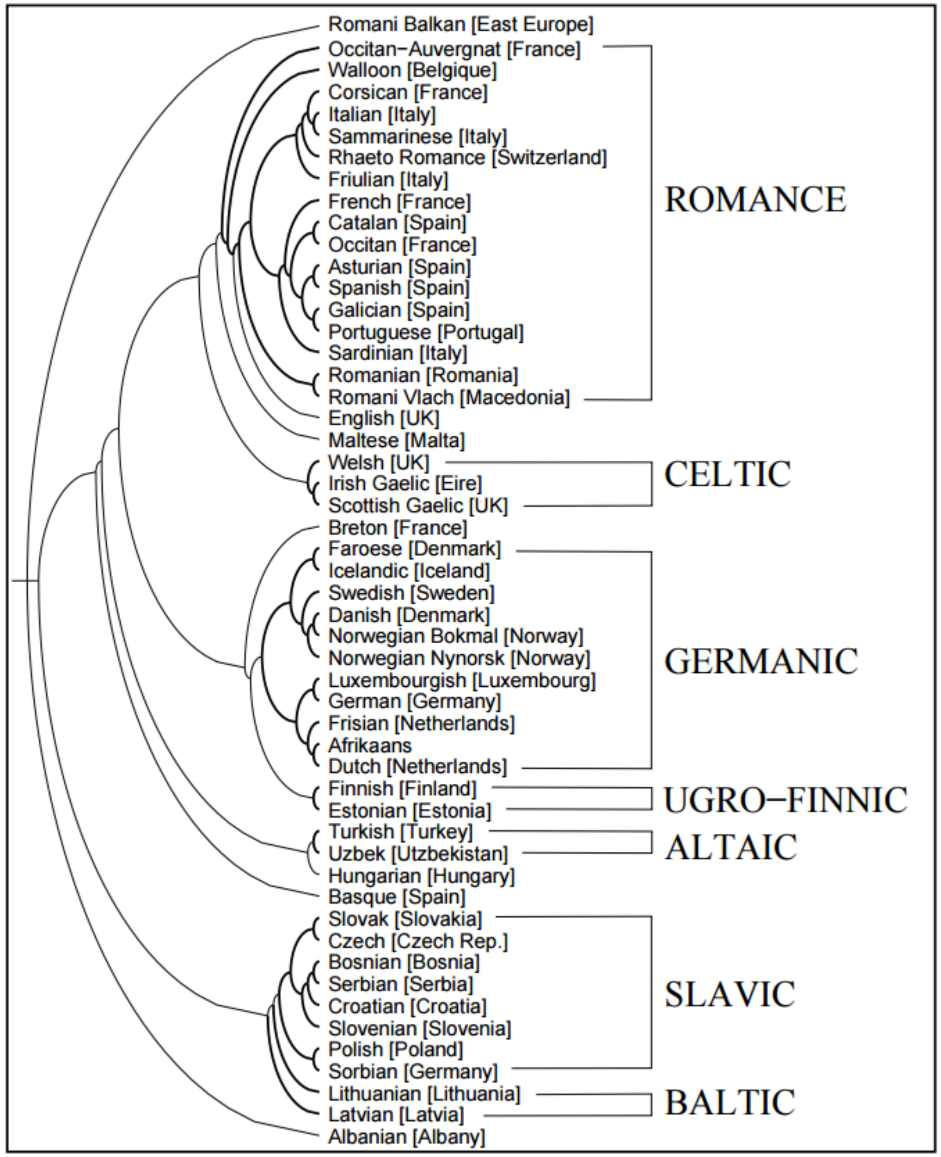
Lempel-Ziv Compression: Drawbacks

- Different features for different ordering
- D1=abcd, D2=ceab, D3=bce
  - Concatenation D1D2D3 yields features {a,b,c,d,ab,bc}
  - Concatenation D2D3D1 yields features {a,b,c,d,ab,ce}
- Performance is sensitive to this ordering



History: Entropy as Compression Difficulty

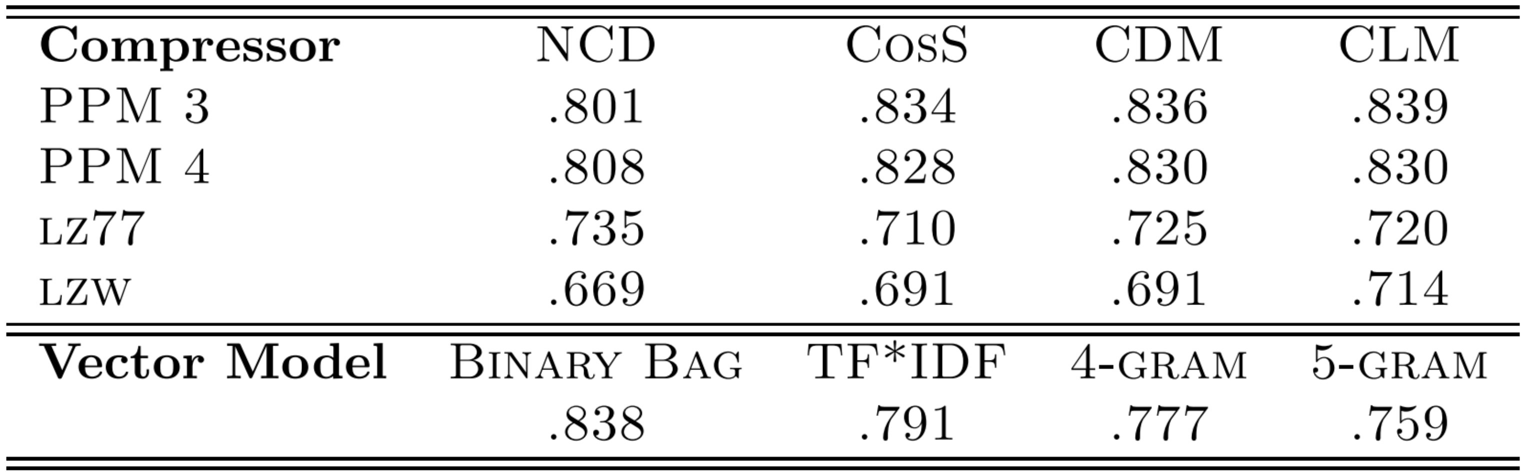
- Benedetto et al. 2001[3] had the idea of computing the 'divergence' between two different text distributions A and B by seeing how hard it is to compress text from distribution B using a compression scheme 'trained' on text from distribution A.
- Used LZ to perform well on an authorship attribution task as well as constructing a language tree of European languages via their translations of *The Universal Declaration of Human Rights*



Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. *Language Trees and Zipping*. PRL 2002.

History: Compression as a Similarity Measure

- Sculley and Brodley 2006[4] show that for several compression schemes the image of the algorithm can be associated to a vector space whose norm is the compression length.
- For each scheme they also define a document similarity measure based on these norms that is used to identify UNIX users based on their user data using Nearest Neighbor classification.



Compressor	NCD	CosS	CDM	CLM
PPM 3	.801	.834	.836	.839
PPM 4	.808	.828	.830	.830
LZ77	.735	.710	.725	.720
LZW	.669	.691	.691	.714
Vector Model	Binary Bag	TF*IDF	4-gram	5-gram
	.838	.791	.777	.759

4 D. Sculley and Carla E. Brodley. *Compression and Machine Learning: A New Perspective on Feature Space Vectors*. DCC 2006.

Compressive Features: Representation

Given a document D:

Set of all possible n-grams: $S = \{x_i \dots x_{i+t-1} : 1 \leq t \leq n, 1 \leq i \leq n - t + 1\}$
Set of all possible pointers: $P = \{(s, l) : s = x_l \dots x_{l+|s|-1}\}$

If all words unique, $|S| = \sum_{k=1}^{n} |D| - k + 1 = n\left(|D| + 1 - \frac{n+1}{2}\right)$. Denote number of pointers as $m = |P| = \mathcal{O}(|S|)$.

Given $w \in \{0,1\}^m$, w *reconstructs* the ith word x_i of D if $\exists\ j$ s.t.

- $w_j = 1$
- the jth pointer (s, l) satisfies $l \leq i < l + |s|$

Represent D by $|D| \times m$ matrix X s.t. $X_{ij} = 1 \iff$ the jth pointer can reconstruct the ith word x_i of D. Then w reconstructs D if $Xw \geq 1$.

Compressive Features: Optimization

Let $d \in \mathbb{R}^m$ be the cost vector of storing the pointers, i.e. $d_i \geq 0$ represents the cost of storing pointer w_i. Let $c(s) \geq 0$ be the cost of storing any n-gram $s \in S$. Let $J(s) \subset \{1, \ldots, m\}$ be the set of all pointers in P sharing string $s \in S$. Find feature representations by solving

$$\min_{w \in \{0,1\}^m} w^T d + \sum_{s \in S} c(s) \|w_{J(s)}\|_\infty \quad \text{subject to} \quad Xw \geq 1$$

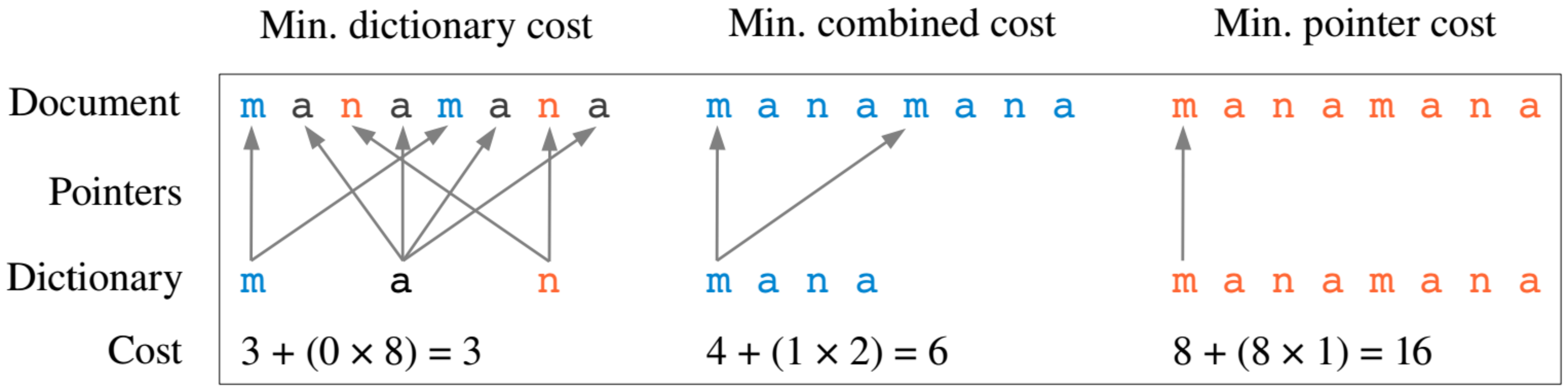
In practice d was set to be a uniform constant $\lambda \in [0, 5]$ and $c(s) = |s|$ (length of the gram) so we get

$$\min_{w \in \{0,1\}^m} \lambda \|w\|_1 + \sum_{s \in S} |s| \|w_{J(s)}\|_\infty \quad \text{subject to} \quad Xw \geq 1$$

To compress a corpus of documents concatenate all of them into one and disallow pointers spanning document boundaries.

Compressive Features: Regularization

The pointer storage cost d can be viewed as a regularization quantifying the trade-off between storing fewer pointers (sparsity) and storing more characters.



Higher pointer cost produces long substrings and tends to hurt accuracy because such substrings occur rarely.

Compressive Features: Convex Relaxation

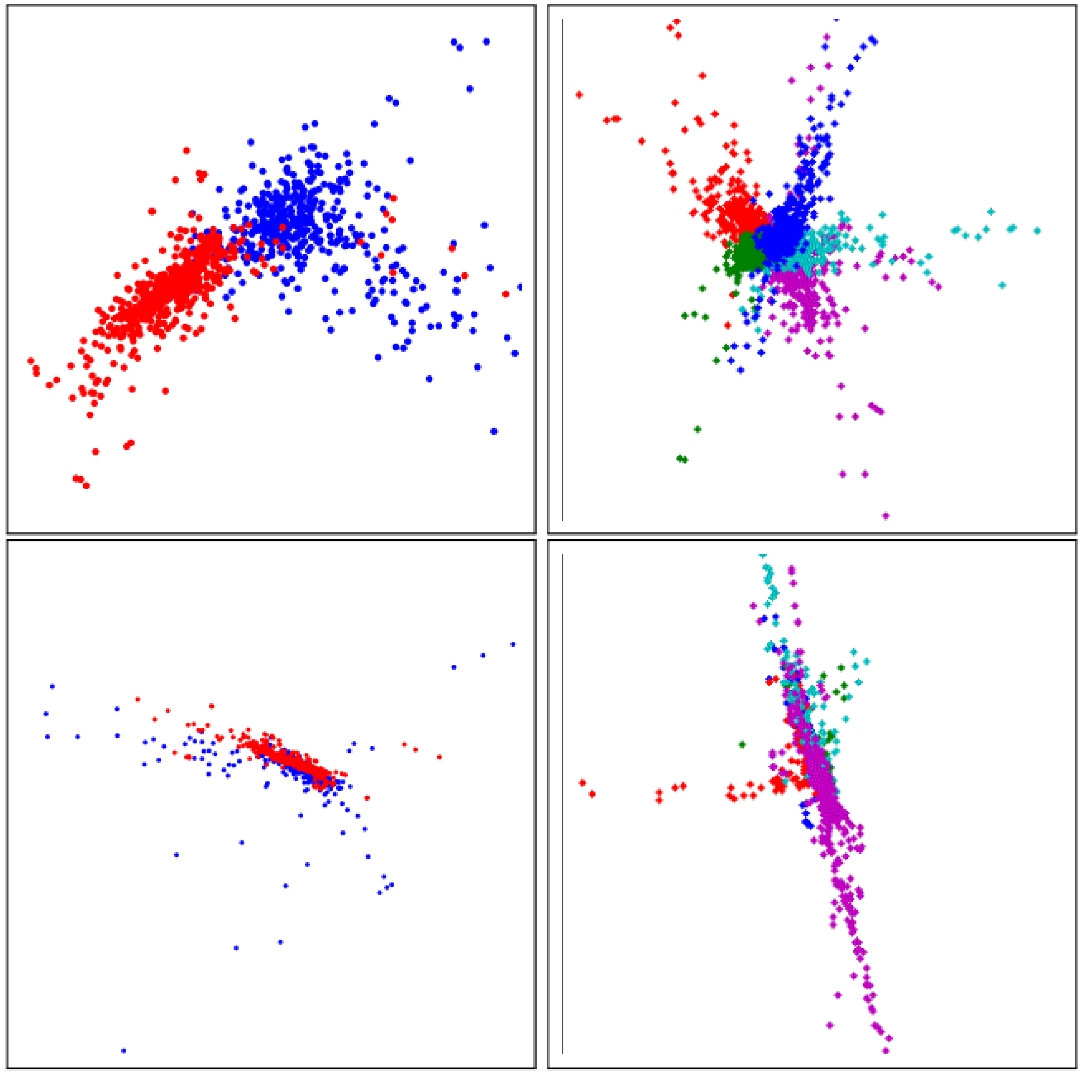
Posed optimization is NP-complete (binary programming). Need to relax to $w \in [0,1]^m$ and solve a series of convex problems

$$\min_{w \in [0,1]^m} w^T d^{(i)} + \sum_{s \in S} c(s) \| D^{(i)}_{J(s)} w_{J(s)} \|_\infty \quad \text{subject to} \quad Xw \geq 1$$

where $D^{(0)}_{jj} = 1$ (identity) and $D^{(i+1)}_{jj} = \max \left\{ 1, \frac{1}{w^{(i)}_j + \varepsilon} \right\}$. This iterative re-weighting up-weights $w^{(i)}_j$ that is not close to binary. Each successive objective is solved by the Alternating Directions Method of Multipliers (ADMM) in which the objective and the constraint are solved iteratively as two coupled optimization problems.

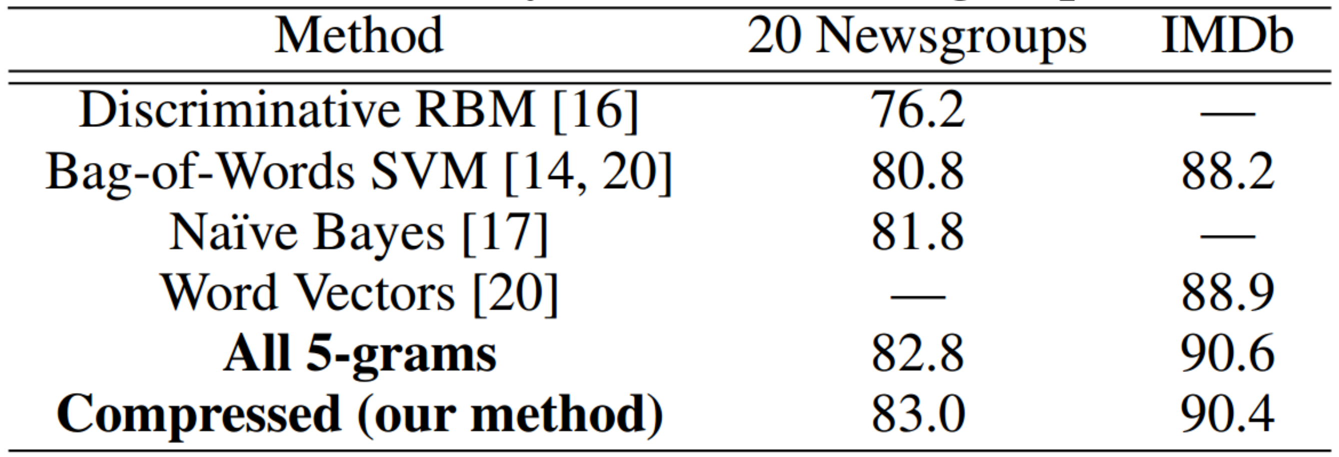
Compressive Features: PCA Clustering

- 2 components from top 10 principal components were picked based on lowest classification error (from 20 news groups corpus) via logistic regression.
- Compressed features display much nicer class structure (top is compressive features, bottom is Bag of 5-gram representations).

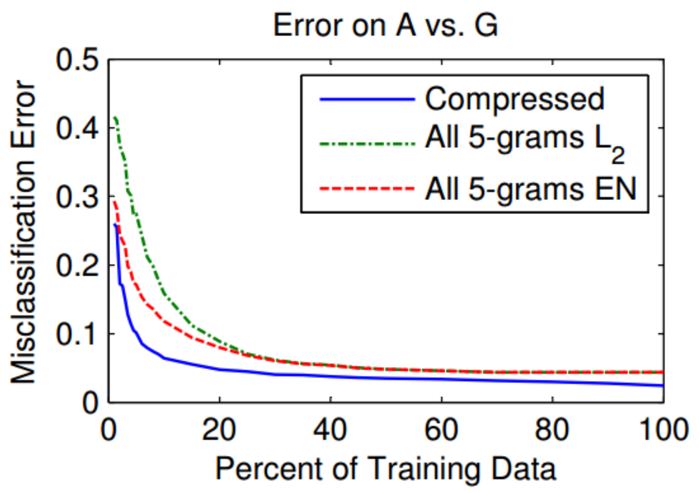


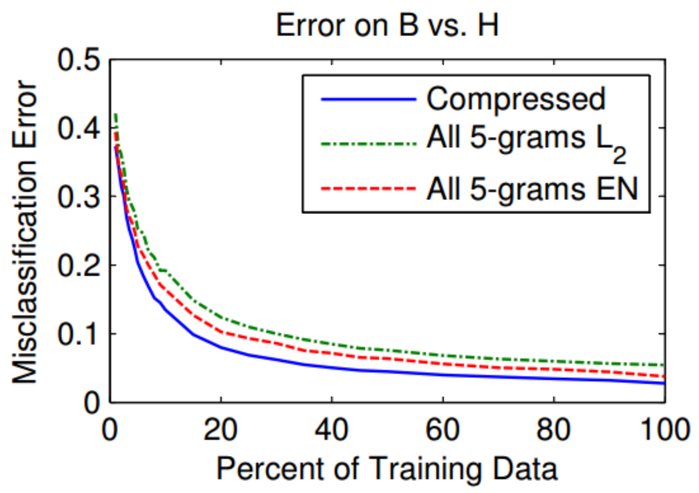
Compressive Features: Text Classification

Classification accuracy on the 20 Newsgroups and IMDb datasets



Method	20 Newsgroups	IMDb
Discriminative RBM [16]	76.2	—
Bag-of-Words SVM [14, 20]	80.8	88.2
Naïve Bayes [17]	81.8	—
Word Vectors [20]	—	88.9
All 5-grams	82.8	90.6
Compressed (our method)	83.0	90.4





Extension: ‘Deep’ Recursive Compression

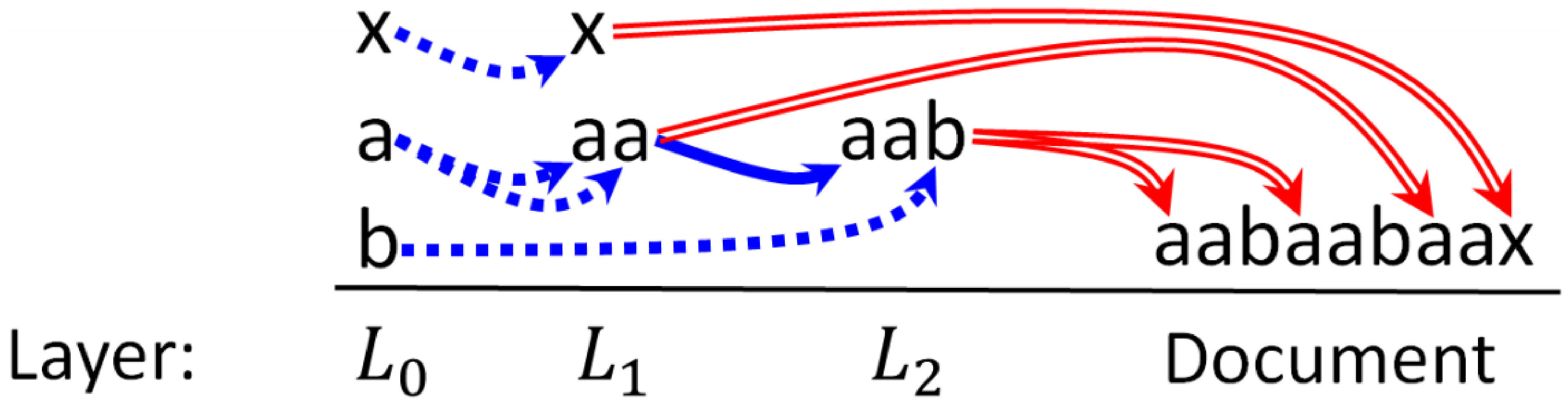
- Compressive feature learning can be extended via recursion – treating the n-grams that compress the documents as documents themselves to be compressed by pointers from shorter n-grams.
- This representation can have many layers, each consisting of sets of (n-gram, pointer) pairs compressing longer n-grams.
- Experimental Results:
  - Trials on small datasets indicates the recursive features are able uncover ‘higher-order structure’ that is useful on tasks such as author identification.
  - However, recursive features do remarkably poorly (10% below SOTA) on a standard sentiment classification task.

Extension: ‘Deep’ Recursive Compression

Feature vectors for recursively compressed documents can be constructed as

$$\hat{X} = X\left(I + \sum_{k=1}^{\infty} G^k\right)$$

where $X \in \mathbb{Z}^{|C|\times|V|}$ is usual featurization (Bag of N-Grams) of the last layer and $G \in \mathbb{Z}^{|V|\times|V|}$ is the weighted adjacency matrix of the DAG of pointers from n-gram to longer n-gram. This is justified via an information flow argument.



Follow-Up Work?

- Issues:
  - Bag of N-Gram vectors are not that hard to store – no performance increase for quite a bit more work
  - Hard to extend to lossy compression – how to define error over natural language (can’t just use Hamming/Euclidean distance)?
- Timing
  - Word embeddings become popular again (Word2Vec – 2013, GloVe – 2014) and found to have nice geometric properties
  - Harder to input compressive features to neural networks