A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors

Mikhail Khodak[*,1], Nikunj Saunshi[*,1], Yingyu Liang[2],
Tengyu Ma[3], Brandon Stewart[1], Sanjeev Arora[1]

1: Princeton University, 2: University of Wisconsin-Madison, 3: FAIR/Stanford University

Motivations

Distributed representations for words / text have had lots of successes in NLP (language models, machine translation, text classification)

Motivations

Distributed representations for words / text have had lots of successes in NLP (language models, machine translation, text classification)

Motivations for our work:

- Can we induce embeddings for all kinds of features, especially those with very few occurrences (e.g. ngrams, rare words)

Motivations

Distributed representations for words / text have had lots of successes in NLP (language models, machine translation, text classification)

Motivations for our work:

- Can we induce embeddings for all kinds of features, especially those with very few occurrences (e.g. ngrams, rare words)
- Can we develop simple methods for unsupervised text embedding that compete well with state-of-the-art LSTM methods

Motivations

Distribute NLP
(language

Motivatio

- Can we th very
  few occ

- Can we develop simple methods for unsupervised text embedding that compete well with state-of-the-art LSTM methods

We make progress on both problems

- Simple and efficient method for embedding features (ngrams, rare words, synsets)

- Simple text embeddings using ngram embeddings which perform well on classification tasks

Word embeddings

- Core idea: Cooccurring words are trained to have high inner product
  - E.g. LSA, word2vec, GloVe and variants

Word embeddings

- Core idea: Cooccurring words are trained to have high inner product
  - E.g. LSA, word2vec, GloVe and variants

- Require few passes over a very large text corpus and do non-convex optimization

corpus  → Optimizing objective → $\left[\; v_w \in \mathbb{R}^d \;\right]$ word embeddings

Word embeddings

- Core idea: Cooccurring words are trained to have high inner product
  - E.g. LSA, word2vec, GloVe and variants

- Require few passes over a very large text corpus and do non-convex optimization

corpus  → Optimizing objective → $v_w \in \mathbb{R}^d$ word embeddings

- Used for solving analogies, language models, machine translation, text classification …

Feature embeddings

- Capturing meaning of other natural language features
  - E.g. ngrams, phrases, sentences, annotated words, synsets

Feature embeddings

- Capturing meaning of other natural language features
  - E.g. ngrams, phrases, sentences, annotated words, synsets

- Interesting setting: features with zero or few occurrences

Feature embeddings

- Capturing meaning of other natural language features
  - E.g. ngrams, phrases, sentences, annotated words, synsets

- Interesting setting: features with zero or few occurrences

- One approach (extension of word embeddings): Learn embeddings for all features in a text corpus

corpus  → Optimizing objective → $\left[\; v_f \in \mathbb{R}^d \; \right]$ feature embeddings

Feature embeddings

Issues

- Usually need to learn embeddings for all features together
  - Need to learn many parameters
  - Computation cost paid is ***prix fixe*** rather than ***à la carte***
- Bad quality for **rare features**

Feature embeddings

Firth revisited: Feature derives meaning from **words** around it

Feature embeddings

Firth revisited: Feature derives meaning from **words** around it

Given a feature f and one (few) context(s) of words around it, can we find a reliable embedding for f efficiently?

Feature embeddings

Firth revisited: Feature derives meaning from **words** around it

Given a feature f and one (few) context(s) of words around it, can we find a reliable embedding for f efficiently?

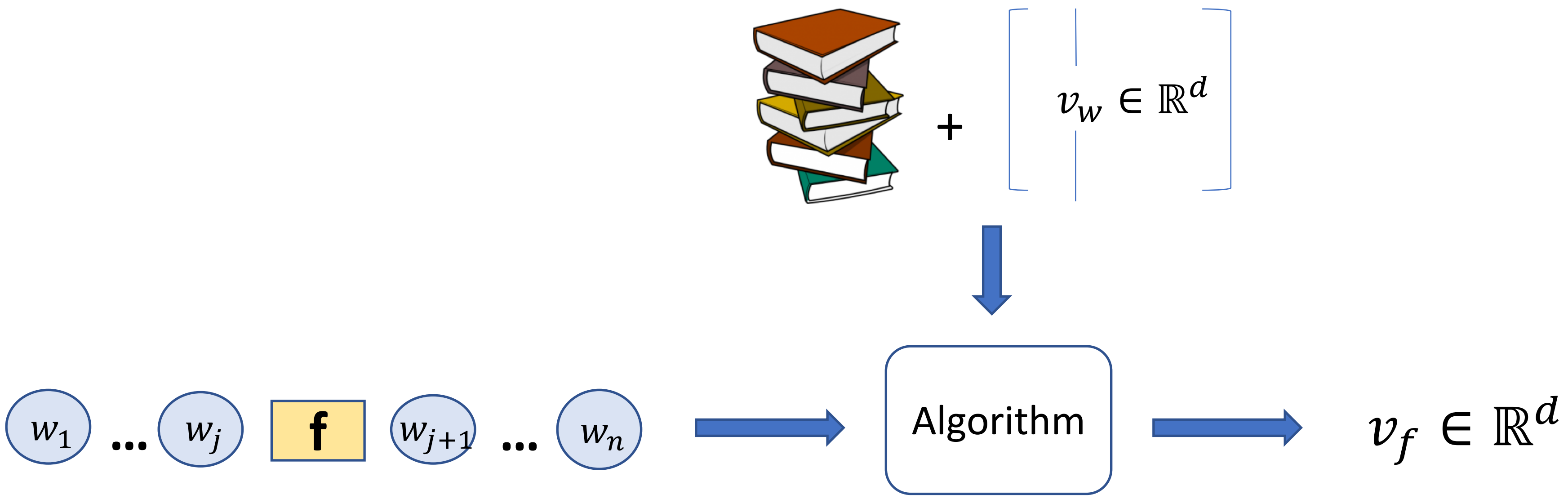
Scientists attending ACL work on **cutting edge** research in NLP

Petrichor: the earthy scent produce when rain falls on dry soil

Roger Federer won the first **set**$^{\mathbf{NN}}$ of the match

Problem setup

Given: Text corpus and high quality word embeddings trained on it



Input: A feature in context(s)

Output: Good quality embedding for the feature

Linear approach

- Given a feature f and words in a context c around it

$$v_f^{avg} = \frac{1}{|c|} \sum_{w \in c} v_w$$

Linear approach

- Given a feature f and words in a context c around it

$$v_f^{avg} = \frac{1}{|c|} \sum_{w \in c} v_w$$

- Issues
  - stop words ("is", "the") are frequent but are less informative
  - Word vectors tend to share common components which will be amplified

Potential fixes

- Ignore stop words

Potential fixes

- Ignore stop words

- SIF weights[1]: Down-weight frequent words (similar to tf-idf)

$$v_f = \frac{1}{|c|} \sum_{w \in c} \alpha_w \, v_w$$

$$\alpha_w = \frac{a}{a + p_w}$$

p_w is frequency of w in corpus

1: Arora et al. ’17

Potential fixes

- Ignore stop words

- SIF weights[1]: Down-weight frequent words (similar to tf-idf)

$$v_f = \frac{1}{|c|}\sum_{w \in c} \alpha_w \, v_w$$

$$\alpha_w = \frac{a}{a + p_w}$$

p_w is frequency of w in corpus

- All-but-the-top[2]: Remove the component of top direction from word vectors

$$v_f = \frac{1}{|c|}\sum_{w \in c} v'_w = (I - uu^T) v_w^{avg}$$

$$u = top\_direction(\{v_w\})$$

$$v'_w = remove\_component(v_w, u)$$

1: Arora et al. '17, 2: Mu et al. '18

Our more general approach

- Down-weighting and removing directions can be achieved by matrix multiplication

$$v_f \approx A \frac{1}{|c|} \sum_{w \in c} v_w = \boxed{A v_f^{avg}}$$

Induced Embedding

Induction Matrix

Our more general approach

- Down-weighting and removing directions can be achieved by matrix multiplication

$$v_f \approx A \frac{1}{|c|} \sum_{w \in c} v_w = \boxed{A v_f^{avg}}$$

Induced Embedding

Induction Matrix

- Learn A by using words as features

$$A^* = argmin_A \sum_w |v_w - A v_w^{avg}|_2^2$$

- Learn A by **linear regression** and is unsupervised

Theoretical justification

- [Arora et al. TACL '18] prove that under a generative model for text, there exists a matrix A which satisfies

$$v_w \approx A v_w^{avg}$$

Theoretical justification

- [Arora et al. TACL '18] prove that under a generative model for text, there exists a matrix A which satisfies

$$v_w \approx A v_w^{avg}$$

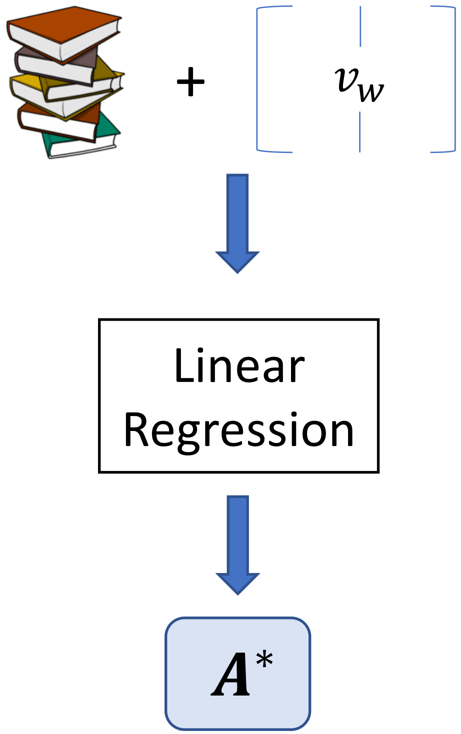
- Empirically we find that the best A^* recovers the original word vectors

$$cosine\left(v_w, A^* v_w^{avg}\right) \geq 0.9$$

A la carte embeddings

1. Learn induction matrix

$$A^* = argmin_A \sum_w |v_w - A v_w^{avg}|_2^2$$



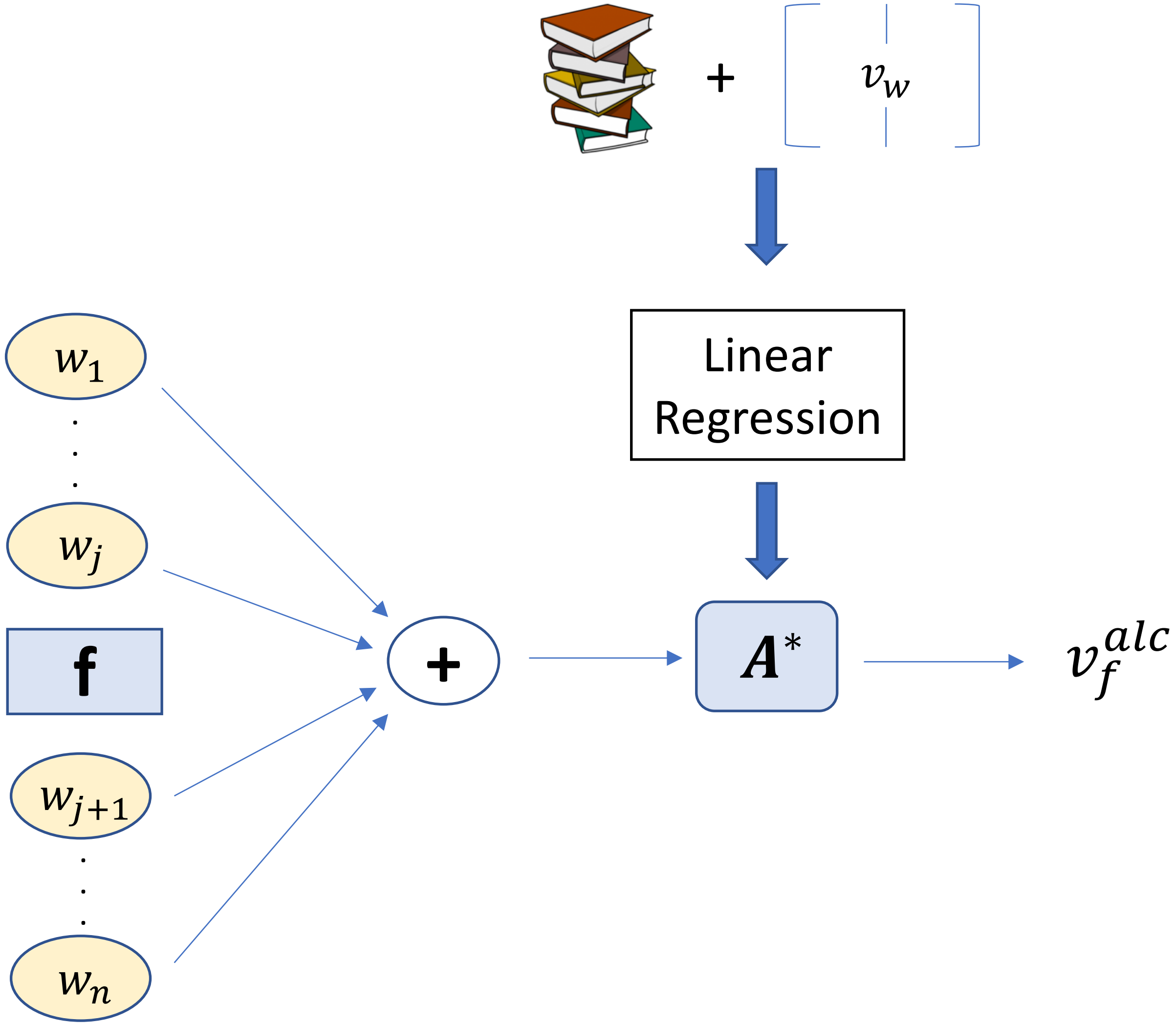
A la carte embeddings

1. Learn induction matrix

$$A^* = argmin_A \sum_w |v_w - A v_w^{avg}|_2^2$$

2. A la carte embeddings

$$v_f^{alc} = A^* v_f^{avg} = A^* \left(\frac{1}{|c|} \sum_{w \in c} v_w \right)$$



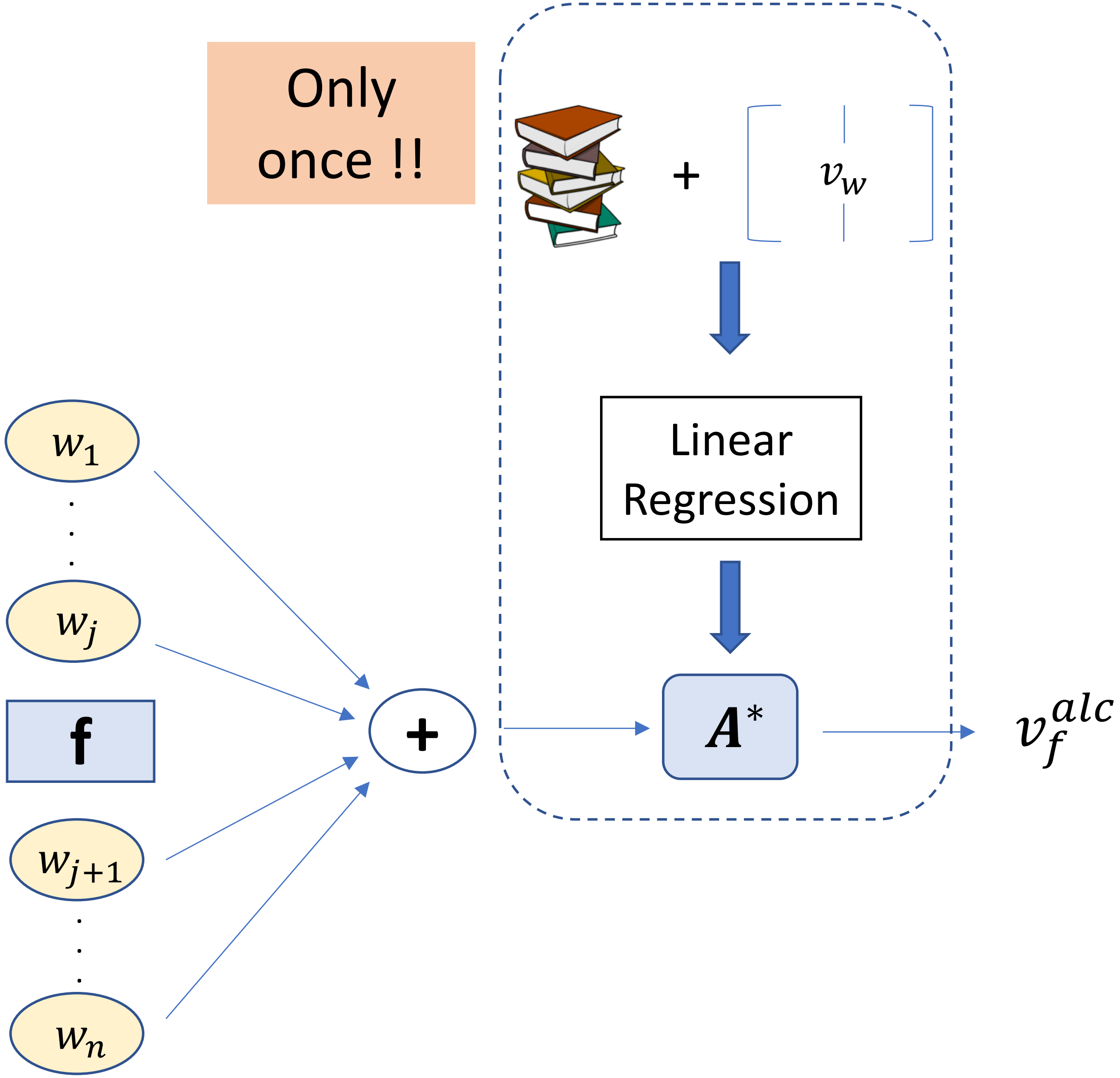
A la carte embeddings

1. Learn induction matrix

$$A^* = argmin_A \sum_w |v_w - Av_w^{avg}|_2^2$$

2. A la carte embeddings

$$v_f^{alc} = A^* v_f^{avg} = A^* \left(\frac{1}{|c|} \sum_{w \in c} v_w \right)$$



Advantages

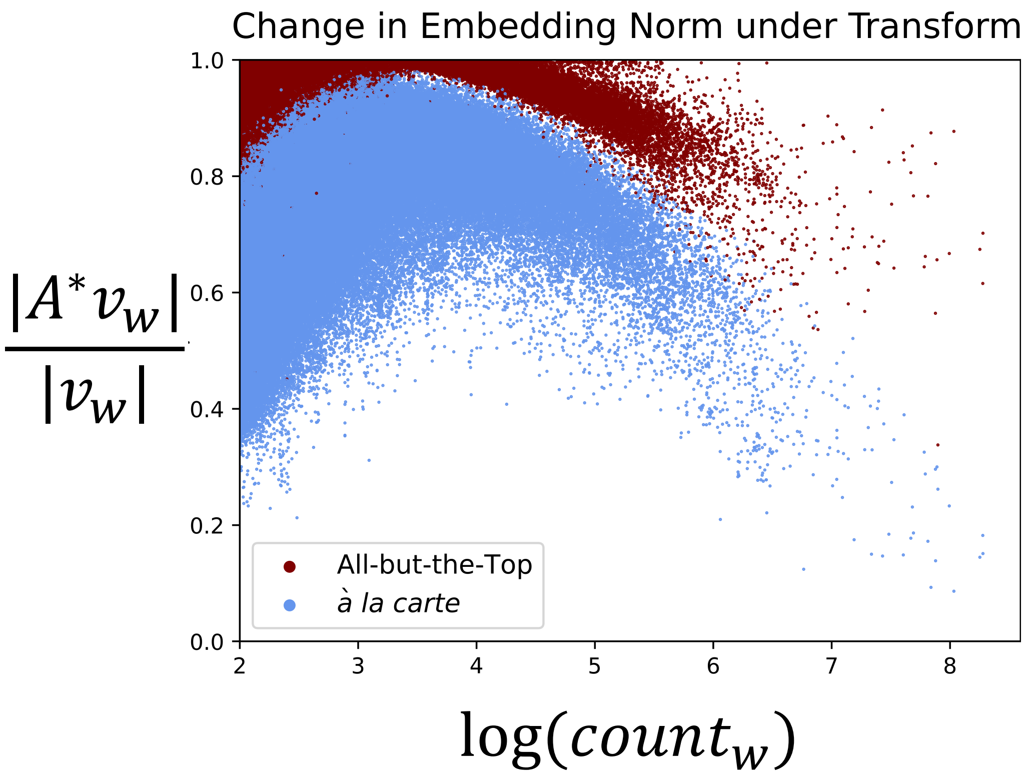
- **à la carte:** Compute embedding only for given feature
- **Simple optimization:** Linear regression
- **Computational efficiency:** One pass over corpus and contexts
- **Sample efficiency:** Learn only d^2 parameters for A^* (rather than Vd)
- **Versatility:** Works for any feature which has at least 1 context

Effect of induction matrix

- We plot the extent to which A^* down-weights words against frequency of words compared to all-but-the-top

Effect of induction matrix

- We plot the extent to which A^* down-weights words against frequency of words compared to all-but-the-top



A^* mainly down-weights words with very high and very low frequency

All-but-the-top mainly down-weights frequent words

Effect of number of contexts

Contextual Rare Words (CRW) dataset[1] providing contexts for rare words

- Task: Predict human-rated similarity scores for pairs of words
- Evaluation: Spearman's rank coefficient between inner product and score

1: Subset of RW dataset [Luong et al. '13]

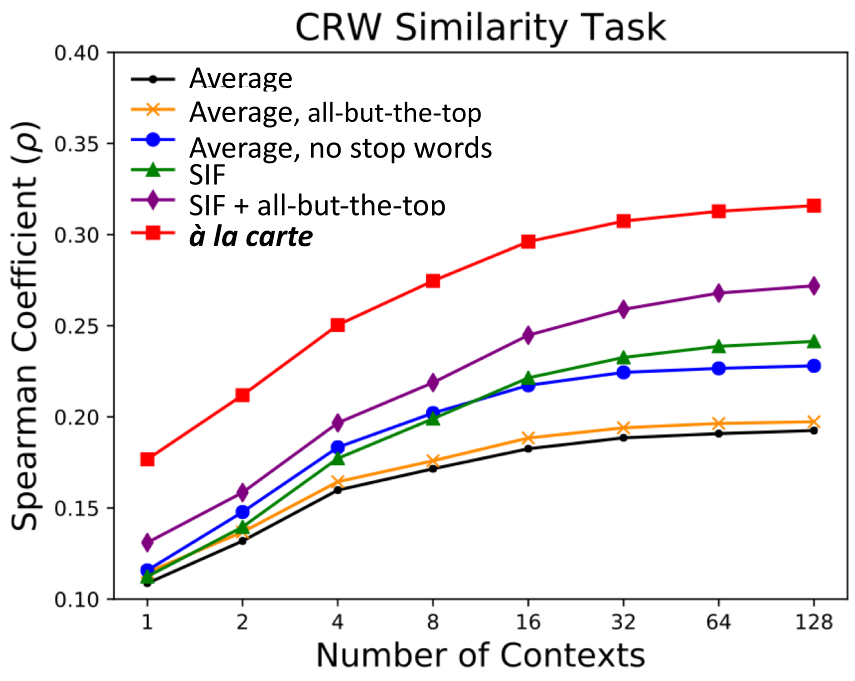
Effect of number of contexts

Contextual Rare Words (CRW) dataset[1] providing contexts for rare words

- Task: Predict human-rated similarity scores for pairs of words
- Evaluation: Spearman's rank coefficient between inner product and score

Compare to the following methods:

- Average of words in context
- Average of non stop words
- SIF weighted average
- all-but-the-top



1: Subset of RW dataset [Luong et al. '13]

Nonce definitional task[1]

- Task: Find embedding for unseen word/concept given its definition
- Evaluation: Rank of word/concept based on cosine similarity with true embedding

iodine: is a chemical element with symbol I and atomic number 53

1: Herbelot and Baroni '17

Nonce definitional task[1]

- Task: Find embedding for unseen word/concept given its definition
- Evaluation: Rank of word/concept based on cosine similarity with true embedding

iodine: is a chemical element with symbol I and atomic number 53

Method	Mean Reciprocal Rank	Median Rank
word2vec	0.00007	111012
average	0.00945	3381
average, no stop words	0.03686	861
nonce2vec[1]	0.04907	623
à la carte	**0.07058**	**165.5**

modified version of word2vec

1: Herbelot and Baroni '17

Ngram embeddings

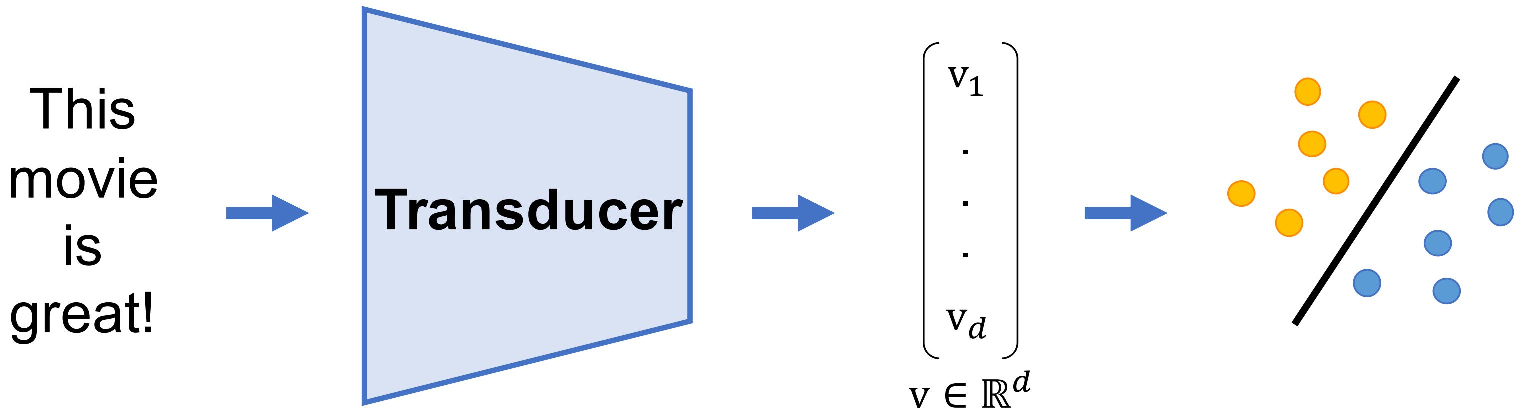
Induce embeddings for ngrams using contexts from a text corpus

We evaluate the quality of embedding for a bigram $f = (w_1, w_2)$ by looking at closest words to this embedding by cosine similarity.

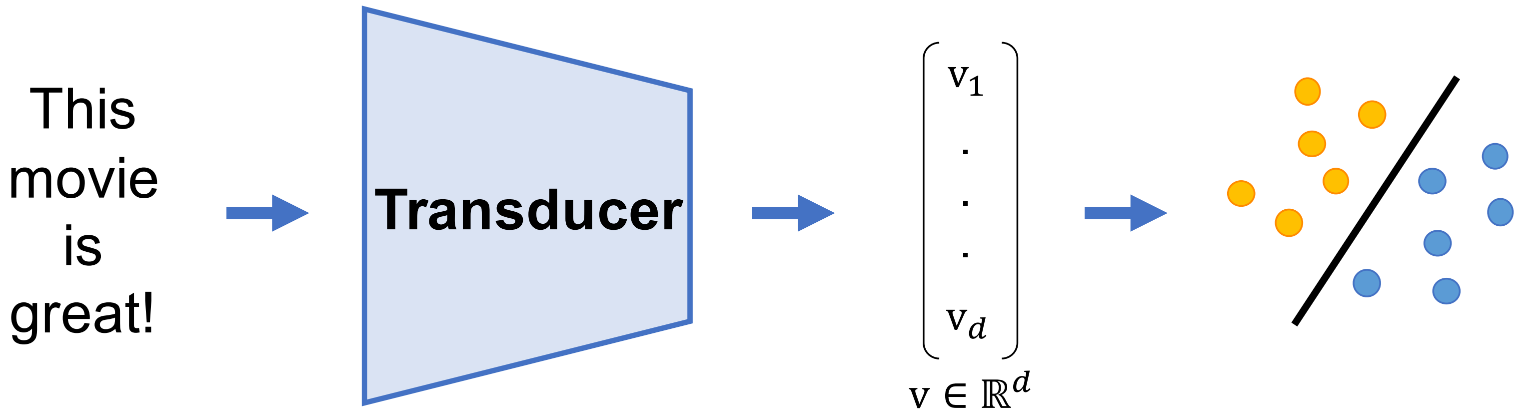
Method	**beef up**	**cutting edge**	**harry potter**	**tight lipped**
$v_f^{add} = v_{w_1} + v_{w_2}$	meat, out	cut, edges	*deathly*, *azkaban*	loose, fitting
v_f^{avg}	but, however	which, both	which, but	but, however
ECO[1]	meats, meat	weft, edges	robards, keach	scaly, bristly
Sent2Vec[2]	*add*, reallocate	*science*, multidisciplinary	naruto, pokemon	wintel, codebase
à la carte ($A^* v_f^{avg}$)	need, *improve*	*innovative*, *technology*	*deathly*, *hallows*	*worried*, very

1: Poliak ’17, 2: Pagliardini et al. ‘18

Unsupervised text embeddings



Unsupervised text embeddings



Sparse

Bag-of-words, Bag-of-ngrams

Good performance

LSTM

Predict surrounding words / sentences

SOTA on some tasks

Linear

Sum of word/ngram embeddings

Compete with Bag-of-ngrams and LSTMs on some tasks

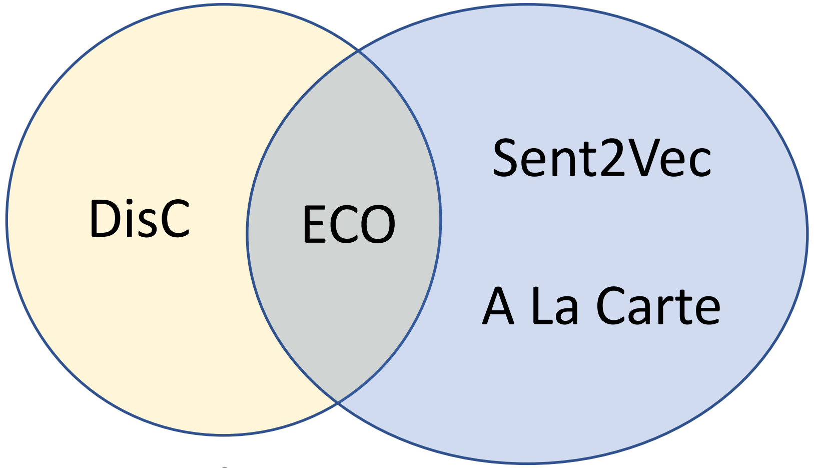
A la carte text embeddings

Linear schemes are typically weighted sums of ngram embeddings

A la carte text embeddings

Linear schemes are typically weighted sums of ngram embeddings

Types of ngrams embeddings



Compositional

Flexible

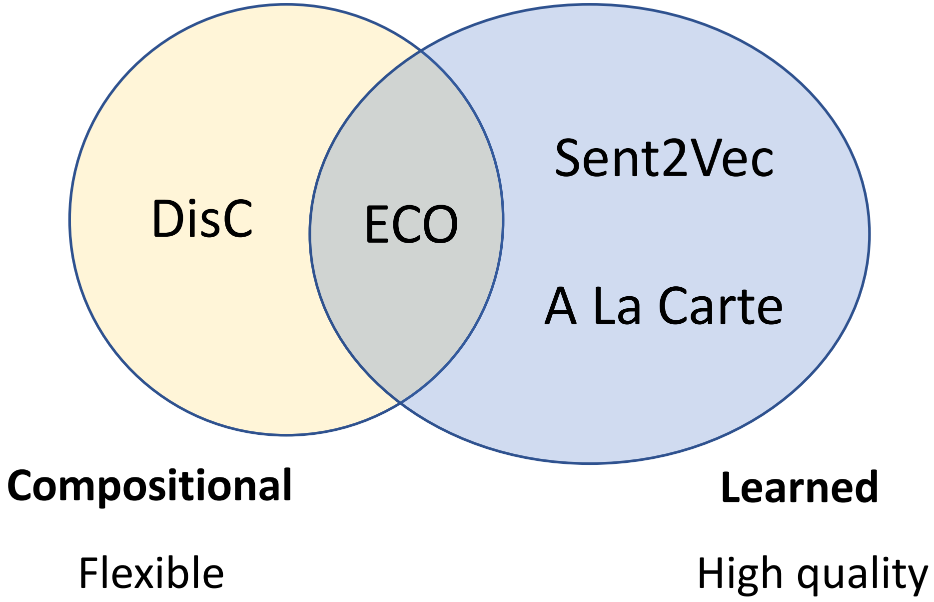
Learned

High quality

A la carte text embeddings

Linear schemes are typically weighted sums of ngram embeddings

Types of ngrams embeddings



A La Carte text embeddings are as concatenations of sum of à la carte ngram embeddings (as in DisC)

$$v_{document}^{n} = \left[\sum v_{word}, \sum v_{bigram}^{alc}, \dots, \sum v_{ngram}^{alc} \right]$$

A la carte text embeddings

$$v_{document}^{alc} = \left[\sum v_{word}, \sum v_{bigram}^{alc}, \dots, \sum v_{ngram}^{alc}\right]$$

	Method	n	dimension	MR	CR	SUBJ	MPQA	TREC	SST (±1)	SST	IMDB
Sparse	Bag-of-ngrams	1-3	100K-1M	77.8	78.3	91.8	85.8	90.0	80.9	42.3	**89.8**
LSTM	Skip-thoughts[1]		4800	80.3	**83.8**	**94.2**	**88.9**	**93.0**	85.1	45.8	
	SDAE[2]		2400	74.6	78.0	90.8	86.9	78.4			
	CNN-LSTM[3]		4800	77.8	82.0	93.6	**89.4**	**92.6**			
	MC-QT[4]		4800	**82.4**	**86.0**	**94.8**	**90.2**	**92.4**	**87.6**		
Linear	DisC[5]	2-3	≤ 4800	80.1	81.5	92.6	87.9	90.0	85.5	**46.7**	89.6
	Sent2Vec[6]	1-2	700	76.3	79.1	91.2	87.2	85.8	80.2	31.0	85.5
	à la carte	2	2400	**81.3**	83.7	93.5	87.6	89.0	**85.8**	**47.8**	**90.3**
		3	4800	**81.8**	**84.3**	**93.8**	87.6	89.0	**86.7**	**48.1**	**90.9**

1: Kiros et al. '15, 2: Hill et al. '16, 3: Gan et al. '17, 4: Logeswaran and Lee '18, 5: Arora et al. '18, 6: Pagliardini et al. '18

Conclusions

- Simple and efficient method for inducing embeddings for many kinds of features, given at least one context of usage
- Embeddings produced are in same semantic space as word embeddings
- Good empirical performance for rare words, ngrams and synsets
- Text embeddings that compete with unsupervised LSTMs

Code is on github: https://github.com/NLPrinceton/ALaCarte
CRW dataset available: http://nlp.cs.princeton.edu/CRW/

Future work

- Zero shot learning of feature embeddings
  - Compositional approaches
- Harder to annotate features (synsets)
- Contexts based on other syntactic structures

Thank you!

Questions?

{nsaunshi, mkhodak}@cs.princeton.edu