Representation Learning and Linguistic Structure

Jacob Eisenstein

Georgia Institute of Technology

July 21, 2016

The deep learning steamroller

- Will the deep learning steamroller flatten NLP?
- If we can just learn the representations we need, what role remains for linguistics?

Before representation learning

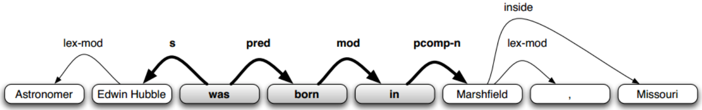
The "NLP stack"

- Read raw text into tokens.
- Tag / chunk / parse text into a **linguistic representation**.
- Extract knowledge, using patterns or features of the linguistic representation.

The NLP stack

Feature type	Left window	NE1	Middle	NE2	Right window
Lexical	[]	PER	[was/VERB born/VERB in/CLOSED]	LOC	[]
Lexical	[Astronomer]	PER	[was/VERB born/VERB in/CLOSED]	LOC	[,]
Lexical	[#PAD#, Astronomer]	PER	[was/VERB born/VERB in/CLOSED]	LOC	[, Missouri]
Syntactic	[]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[]
Syntactic	[Edwin Hubble $\Downarrow_{lex-mod}$]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[]
Syntactic	[Astronomer $\Downarrow_{lex-mod}$]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[]
Syntactic	[]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[$\Downarrow_{lex-mod}$,]
Syntactic	[Edwin Hubble $\Downarrow_{lex-mod}$]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[$\Downarrow_{lex-mod}$,]
Syntactic	[Astronomer $\Downarrow_{lex-mod}$]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[$\Downarrow_{lex-mod}$,]
Syntactic	[]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[$\Downarrow_{inside}$ Missouri]
Syntactic	[Edwin Hubble $\Downarrow_{lex-mod}$]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[$\Downarrow_{inside}$ Missouri]
Syntactic	[Astronomer $\Downarrow_{lex-mod}$]	PER	[$\Uparrow_s$ was $\Downarrow_{pred}$ born $\Downarrow_{mod}$ in $\Downarrow_{pcomp-n}$]	LOC	[$\Downarrow_{inside}$ Missouri]

Table 3: Features for ‘Astronomer Edwin Hubble was born in Marshfield, Missouri’.



Pipeline pros and cons

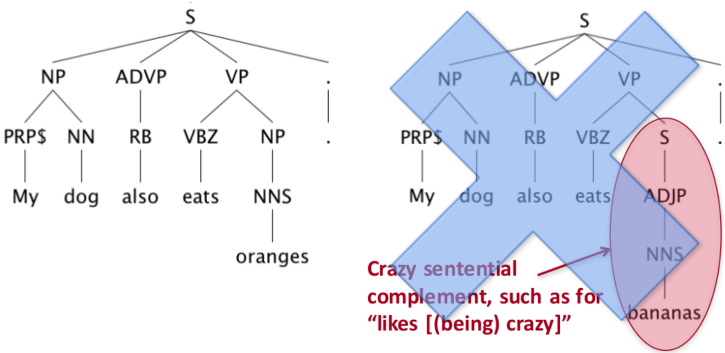
- ☺ Linguistic theories and ontologies can be encoded into features.
- ☺ Intermediate representations are interpretable and generalizable across tasks.
- ☺ Can train systems using generic and robust machine learning methods (e.g. structured perceptron)

Pipeline pros and cons

- ☺ Linguistic theories and ontologies can be encoded into features.
- ☺ Intermediate representations are interpretable and generalizable across tasks.
- ☺ Can train systems using generic and robust machine learning methods (e.g. structured perceptron)

- ☹ Data hungry, yielding poor performance on the “long tail”.
- ☹ Hard to port to new languages, genres, registers ....

Symbolic representations are brittle!

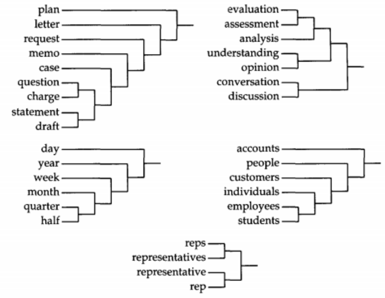


(Figure from Chris Manning's 2016 SIGIR keynote)

Representation learning to the rescue?

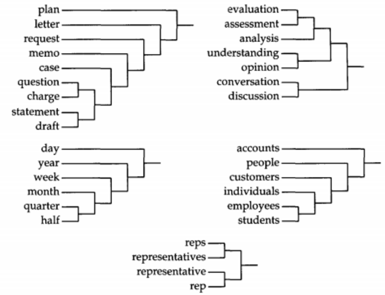
- **Motivating observation**: discrete “one-hot” linguistic representations are too sparse and too high-dimensional to support generalization. Errors cascade through the pipeline!
- **But!** We can learn better representations from unlabeled data.
- NB: this is a very old idea, e.g., latent semantic analysis, Brown clusters, topic models, Bayesian latent variable models, ...

Representation Learning (cuddly, non-threatening version)

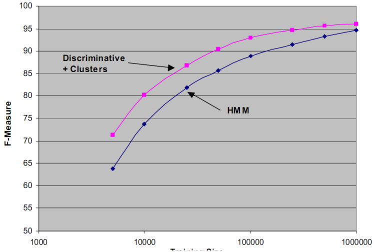


Brown clusters...

Representation Learning
(cuddly, non-threatening version)



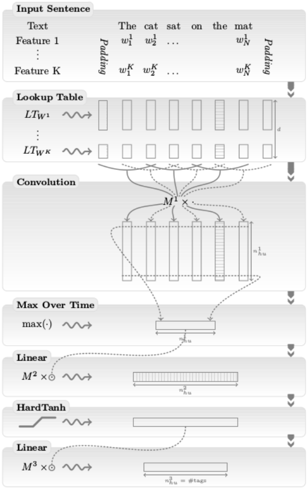
Brown clusters...



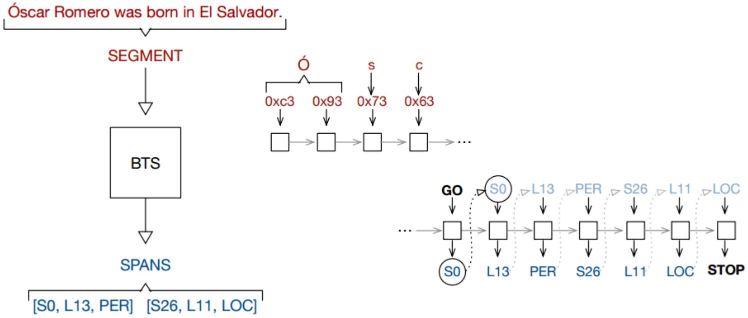
... dramatically improve named entity tagging! (Miller et al, 2004)

Representation Learning (steamroller version)

- "Natural Language Processing from Scratch" (Collobert and Weston 2008, Collobert et al. 2011)
- Stated goal: NLP with as little linguistic knowledge as possible!



Representation Learning (velociraptor version)



(Gillick et al., NAACL 2016)

Representation learning pros and cons

- ☺ Distributional hypothesis: we can learn word representations from unlabeled data.
- ☺ Unsupervised learning has gotten way better.
- ☺ Good empirical results in new domains and in low-resource languages.

Representation learning pros and cons

- ☺ Distributional hypothesis: we can learn word representations from unlabeled data.
- ☺ Unsupervised learning has gotten way better.
- ☺ Good empirical results in new domains and in low-resource languages.

- ☹ Distributional information can't help with unseen words and other phenomena
- ☹ Distributed (vector) representations are reasonable for words, maybe. But for phrases? Sentences? Documents?

The big picture

DTRA has a vision in which NLP systems extract and assemble evidence about WMD proliferation events.

- How is this software going to be built?
- Will its internal representations be learned from data?
- Or will they be based on linguistic theory?

The big picture

DTRA has a vision in which NLP systems extract and assemble evidence about WMD proliferation events.

- How is this software going to be built?
- Will its internal representations be learned from data?
- Or will they be based on linguistic theory?

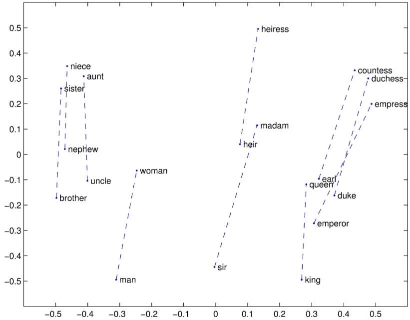
Our view: both are necessary, but more research is needed on how linguistic theory and representation learning can best be combined.

Design dimensions for word embeddings

- Discrete vs continuous
- Recursive vs linear
- Compositional vs distributional

Word embeddings

- Word embeddings can capture lexical semantic properties.
- This works because lexical semantics correlates with distributional statistics.



RNN Word Embeddings

- Each word w has an embedding $\boldsymbol{x}_w$.
- Mikolov et al (2010): estimate these embeddings by making them into parameters of a language model.

$$\boldsymbol{h}_t = \text{Sigmoid}(\Theta \boldsymbol{h}_{t-1} + \boldsymbol{x}_{w_t}) \qquad (1)$$

$$y_{t+1} \sim \text{SoftMax}(\Phi \boldsymbol{h}_t) \qquad (2)$$

- Estimation maximizes the likelihood $P(y_1, y_2, \ldots, y_T; \Theta, \Phi, \boldsymbol{x})$.
- Word2vec, GLoVE, bilinear language models are very similar.

Design dimensions for word embeddings

- Discrete vs continuous
- Recursive vs linear
- Compositional vs distributional

Design dimensions for word embeddings

- Discrete vs **continuous**
- Recursive vs **linear**
- Compositional vs **distributional**

Design dimensions for word embeddings

- Discrete vs **continuous**
- **Recursive** vs linear
- Compositional vs **distributional**

Levy and Golberg (2014): dependency-based word embeddings

Design dimensions for word embeddings

- Discrete vs **continuous**
- Recursive vs **linear**
- **Compositional** vs distributional

Pros/cons of distributional embeddings

Word embeddings like `word2vec`:

- work great for common words with large counts: king, queen, man, woman, . . . ;
- work well for morphologically simple languages, because the type-token ratio is small;
- not shown to work for rare words: even when embeddings are available, they are unreliable if based on sparse counts.

Compositional word embeddings

Idea: build word representations **compositionally**!

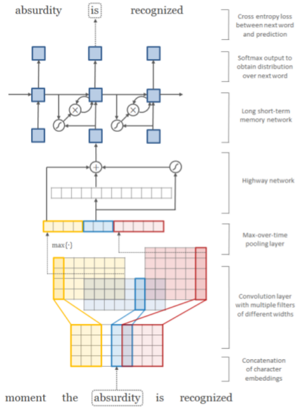
- Induce embeddings of subword units: characters or morphemes.
- Then learn to compose these subwords embeddings into word embeddings.

Why?

- Share information across structurally related words, which can help in the long tail (e.g., *perspicuous, perspicuity*).
- Better performance on morphologically rich languages.

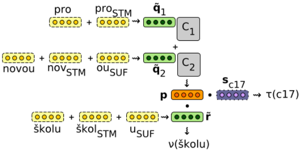
Word embeddings from characters

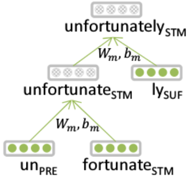
- The compositional character model (Ling et al 2015) is a recurrent neural network on characters.
- The **character-aware language model** (Kim et al 2016) is a convolutional network on characters.



Word embeddings from morphemes

- Botha and Blunsom (2014): word representations are a sum of morpheme embeddings (SUMEMBED)
- Luong et al (2013): word representations are computed recursively over a morphological parse.





Word embeddings from A to Z

Character and morpheme embeddings are promising steps forward, **but**:

1. The mapping from character n-grams to meaning is complex:
   - invisible: *not* visible
   - inflammable: *very* flammable

   Distributional information should override morphology when counts are sufficient.
2. Morphology and lexical semantics are often discrete: animacy, number, etc.

Design dimensions for word embeddings

- Discrete vs **continuous**
- Recursive vs **linear**
- **Compositional** vs distributional

Design dimensions for word embeddings

- **Discrete** vs **continuous**
- Recursive vs **linear**
- **Compositional** vs **distributional**

Latent boolean word embeddings

Our solution: encode word meaning as a boolean **latent variable** $\boldsymbol{b}$ in a probabilistic model.

- Distributional statistics $\boldsymbol{x}_{1:N}$ are an **emission** from the likelihood $P(\boldsymbol{x} \mid \boldsymbol{b})$.
- Morphological information $\boldsymbol{u}$ informs a **prior** distribution $P(\boldsymbol{b} \mid \boldsymbol{u})$.
- Posterior **beliefs** about word meaning are encoded in a variational distribution $q(\boldsymbol{b})$, which is a distributed representation.

Latent boolean word embeddings

Advantages of this formulation:

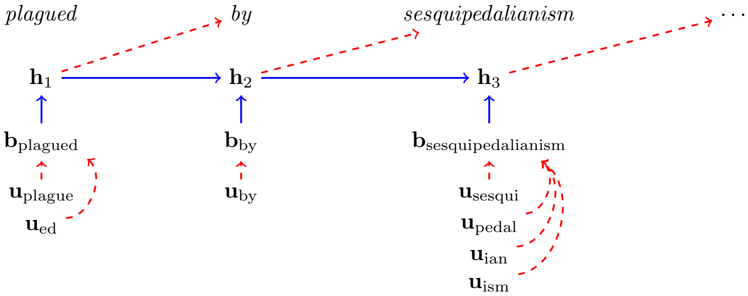
- Naturally interpolates between distributional and morphological information:

$$P(\boldsymbol{b} \mid \boldsymbol{x}, \boldsymbol{u}) \propto P(\boldsymbol{x} \mid \boldsymbol{b}) P(\boldsymbol{b} \mid \boldsymbol{u})$$

- Given partial knowledge of lexical semantics (e.g., wordnet), latent variables can be "clamped" to true values.

Challenge: integrating neural network and Bayesian latent variables in a single end-to-end architecture.

RNN Formulation



$$b^{(i)}_{\text{plagued}} \sim \text{Bernoulli}(\sigma(u^{(i)}_{\text{plague}} + u^{(i)}_{\text{ed}})) \tag{3}$$

$$\boldsymbol{h}_2 = \text{Sigmoid}(\Theta \boldsymbol{h}_1 + \boldsymbol{b}_{\text{by}}) \tag{4}$$

$$x_3 \sim \text{Multinomial}(\text{SoftMax}[\mathbf{V} h_2]) \tag{5}$$

Objective function

The log probability of the corpus is:

$$\begin{aligned}
\log P(x) &= \log \sum_{\boldsymbol{b}} P(\boldsymbol{x}, \boldsymbol{b}) \\
&= \log \sum_{\boldsymbol{b}} P(x \mid \boldsymbol{b}) P(\boldsymbol{b}) \\
&= \log \sum_{\boldsymbol{b}} \frac{Q(\boldsymbol{b})}{Q(\boldsymbol{b})} P(\boldsymbol{x} \mid \boldsymbol{b}) P(\boldsymbol{b}) \\
&\geq E_q[\log P(\boldsymbol{x} \mid \boldsymbol{b})] + E_q[\log P(\boldsymbol{b})] \\
&\quad - E_q[\log Q(\boldsymbol{b})]
\end{aligned}$$

Here we use Jensen's inequality to formulate a variational lower bound on the marginal log-likelihood $\log P(x)$.

Bayesian reasoning

We represent our posterior beliefs about $\boldsymbol{b}$ through a variational mean-field approximation,

$$Q(\boldsymbol{b}) = \prod_{w} q_w(\boldsymbol{b}_w)$$

$$q_w(\boldsymbol{b}_w) = \prod_{j} q_{w,j}(b_{w,j}).$$

Due to the variational approximation, we call this approach VarEmbed.

Implementation

- We use the long short-term memory (LSTM) variant of RNNs to avoid the vanishing gradient problem.
- Point estimates for latent state:

$$
\begin{aligned}
E_{q_{1:n}}[\boldsymbol{h}_n] =& E_{q_{1:n}}[f(\boldsymbol{\Theta}^{(w)}\boldsymbol{h}_{n-1} + \boldsymbol{b}_{w_n})] \\
\approx& f(\boldsymbol{\Theta}^{(w)} E_{q_{1:n-1}}[\boldsymbol{h}_{n-1}] + E_{q_n}[\boldsymbol{b}_{w_n}])
\end{aligned}
$$

- Our `Blocks` implementation takes ~ 20 hours to train on 22 million tokens, using a commodity gaming GPU.

Relationship to SumEmbed

The point estimate approximation implies:

$$
\begin{aligned}
L &\geq \sum_{t}^{N} E_q[\log P(x_t \mid \boldsymbol{x}_{1:t-1}; \boldsymbol{b})] + E_q[\log P(\boldsymbol{b})] - E_q[\log Q(\boldsymbol{b})] \\
&\approx \sum_{t}^{N} \log P(x_t \mid \boldsymbol{x}_{1:t-1}; E_q[\boldsymbol{b}]) + E_q[\log P(\boldsymbol{b})] - E_q[\log Q(\boldsymbol{b})] \\
&= \sum_{t}^{N} \log P(x_t \mid \boldsymbol{x}_{1:t-1}; E_q[\boldsymbol{b}]) + KL(Q(\boldsymbol{b})||P(\boldsymbol{b})),
\end{aligned}
$$

where $KL(Q(\boldsymbol{b})||P(\boldsymbol{b}))$ is the KL-divergence.

Relationship to SumEmbed

The point estimate approximation implies:

$$\begin{aligned}
L &\geq \sum_{t}^{N} E_q[\log P(x_t \mid \boldsymbol{x}_{1:t-1}; \boldsymbol{b})] + E_q[\log P(\boldsymbol{b})] - E_q[\log Q(\boldsymbol{b})] \\
&\approx \sum_{t}^{N} \log P(x_t \mid \boldsymbol{x}_{1:t-1}; E_q[\boldsymbol{b}]) + E_q[\log P(\boldsymbol{b})] - E_q[\log Q(\boldsymbol{b})] \\
&= \sum_{t}^{N} \log P(x_t \mid \boldsymbol{x}_{1:t-1}; E_q[\boldsymbol{b}]) + KL(Q(\boldsymbol{b})||P(\boldsymbol{b})),
\end{aligned}$$

where $KL(Q(\boldsymbol{b})||P(\boldsymbol{b}))$ is the KL-divergence.

- The SumEmbed objective is the same, but without the KL-divergence term!

Results: Word similarity

	word2vec	SumEmbed	VarEmbed
Wordsim353			
all words (353)	n/a	42.9	**48.8**
in-vocab (348)	**51.4**	45.9	51.3
rare words (rw)			
all words (2034)	n/a	23.0	**24.0**
in-vocab (715)	33.6	37.3	**44.1**

Table: Word similarity evaluation results, as measured by Spearmann's $\rho \times 100$. `word2vec` cannot be evaluated on all words, because embeddings are not available for out-of-vocabulary words. The total number of words in each dataset is indicated in parentheses.

Results: Lexical semantics

	all words (4199)	in vocab (3997)
word2vec	n/a	**34.8**
SumEmbed	32.8	33.5
VarEmbed	**33.6**	34.7
morphemes only		
SumEmbed	24.7	25.1
VarEmbed	**30.2**	**31.0**

Table: Alignment with lexical semantic features (Wordnet supersenses), as measured by QVEC (Tsvetkov et al., 2015). Higher scores are better, with a maximum possible score of 100.

Results: Part-of-speech tagging

	dev	test
`word2vec`	92.42	92.40
SumEmbed	**93.26**	**93.26**
VarEmbed	93.05	93.09

Table: Part-of-speech tagging accuracies

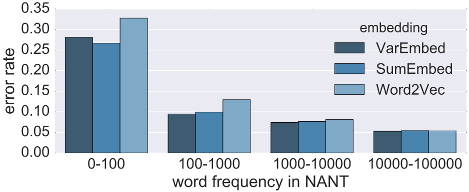


Figure: Error rates by word frequency.

Design dimensions for word embeddings

- **Discrete** vs **continuous**
- Recursive vs **linear**
- **Compositional** vs **distributional**

Discussion

VarEmbed is a more flexible integration of linguistic structure and representation learning...

Discussion

VarEmbed is a more flexible integration of linguistic structure and representation learning... but the linguistic structure is minimal.

Discussion

VarEmbed is a more flexible integration of linguistic structure and representation learning... but the linguistic structure is minimal.

- How to extend this approach to richer models of morphology?
- Can we incorporate existing (or forthcoming!) lexical semantic resources?
- How to integrate variational word embeddings in sentence-level and document-level text analysis?